DSP Benchmark Results for the Latest VLIW-Based Processors

Berkeley Design Technology, Inc.

www.BDTI.com

1

Outline

- VLIW basics and a case study
  - What's VLIW?
  - Why VLIW?
  - The TMS320C62xx
  - Advantages, disadvantages of VLIW
- Other VLIW DSP architectures
  - StarCore SC140
  - Infineon Carmel
  - TI TMS320C64xx
  - TI TMS320C55xx
- Benchmark results and analysis

2

Multi-Issue Architectures
A Break From the Past for DSPs

- Until ~1997, most DSPs were very similar
  - Specialized execution units
  - Specialized, complex instruction sets
    - Difficult to program in assembly
    - Unfriendly compiler targets
  - One instruction per instruction cycle
- Multi-issue architectures are different
  - Multiple independent instructions per cycle
  - Often, simple, orthogonal instruction sets

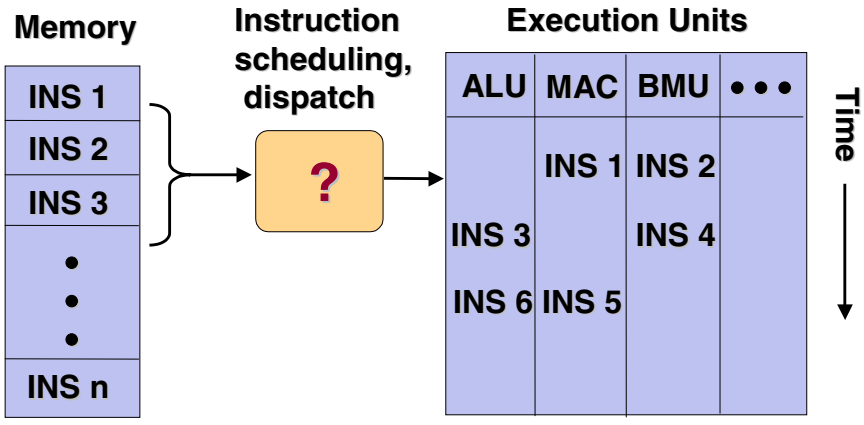
Multi-Issue DSP Architectures: Why?

- Big performance boost, plus:
  - More regular architecture for better compilability
    - RISC-based approach
      - E.g., TMS320C62xx, SC140

      - or -

  - Maintain compatibility
    - Probably not RISC
      - E.g., TMS320C55xx

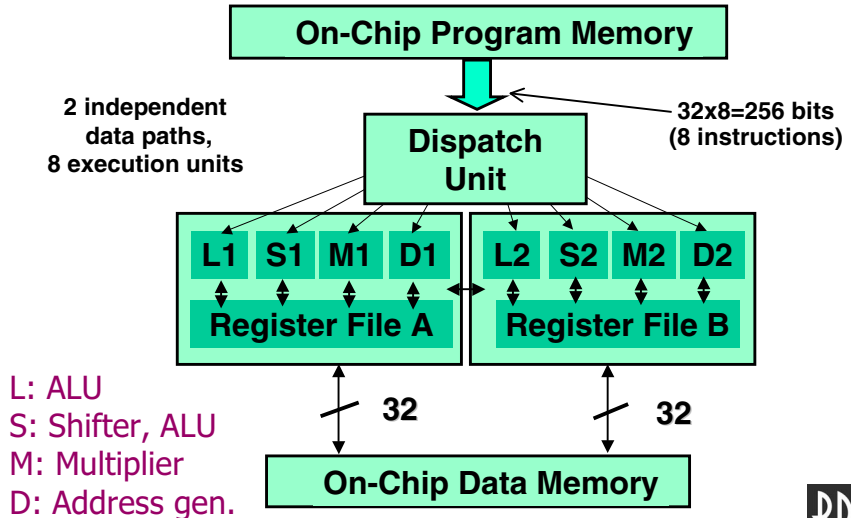
VLIW vs Superscalar



Characteristics of VLIW Processors

- Multiple independent instructions per cycle, packed into single large "instruction word" or "packet"
  - Execution unit destination for each sub-instruction may be implied by instruction position, or may be encoded within each sub-instruction
- Large complement of independent execution units
- In most cases: more regular, orthogonal, RISC-like instructions
  - Usually wider than typical DSP instructions
  - Usually simpler than typical DSP instructions
  - Large, uniform register sets
- Wide program and data buses

Example VLIW DSP: The TI TMS320C62xx



Advantages of VLIW Architectures

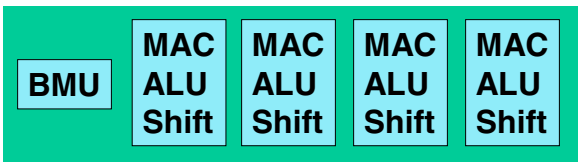
- Increased performance
- Better compiler targets
- Potentially easier to program
- Potentially scalable
  - Can add more execution units, allow more instructions to be packed into the VLIW instruction
  - Or remove duplicated execution units, to reduce cost

Challenges for VLIW Architectures

- New kinds of programmer/compiler complexity
  - Programmer (or code-generation tool) must optimize instruction scheduling
  - Deep pipelines and long latencies can be confusing, may make peak performance elusive
- Increased memory use
  - High program memory bandwidth requirements
- Often, poor energy efficiency
- Misleading MIPS ratings

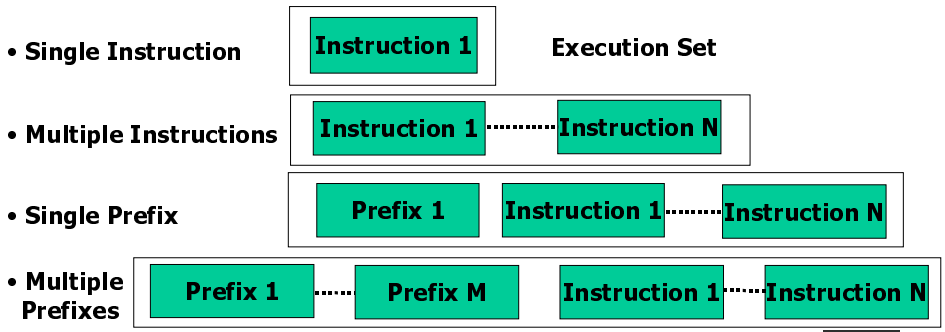
StarCore SC140 Core
The First Implementation of the SC100

- Up to 6 instructions per cycle
  - Fewer than 'C62xx, but more powerful
    - 4 MACs/cycle vs 2 for 'C62xx
- Short (5-stage) pipeline
- Current development chip operates at 300 MHz at 1.5 volts



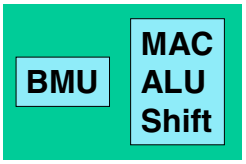
StarCore SC140

- Uses 16-bit instructions with optional 16-bit prefixes ("VLES")
  - Results in very good code density
  - Prefixes used for, e.g., conditional execution, extended register options



StarCore SC110 Core
A Scaled-Down VLIW Core

- Reduced-cost SC100 family member
- Up to 3 instructions per cycle
  - One MAC/ALU/Shift, two AGU
- Narrower buses than SC140
  - 64 bits vs 128 bits in SC140
- Projected 300 MHz at 1.5 volts
- Targets DSL modems, wireless handsets, IP telephony, automotive, consumer electronics



Infineon Carmel

- 16-bit fixed-point VLIW DSP core from Infineon (Siemens)
  - In silicon at 180 MHz, 0.18 µm
- Two data paths, six execution units

- Mixed-width 24/48-bit instruction set
- Can execute in parallel:
  - One 48-bit instruction, or
  - One or two 24-bit instructions, or
  - Up to six instructions as part of a "CLIW"

Infineon Carmel
CLIW (Configurable Long Instruction Word)

General format:

```
cliw name (operand1, ... , operand 4) {
  ALU1 || MAC1 || ALU2 || MAC2 || MOV1 || MOV2
}
```

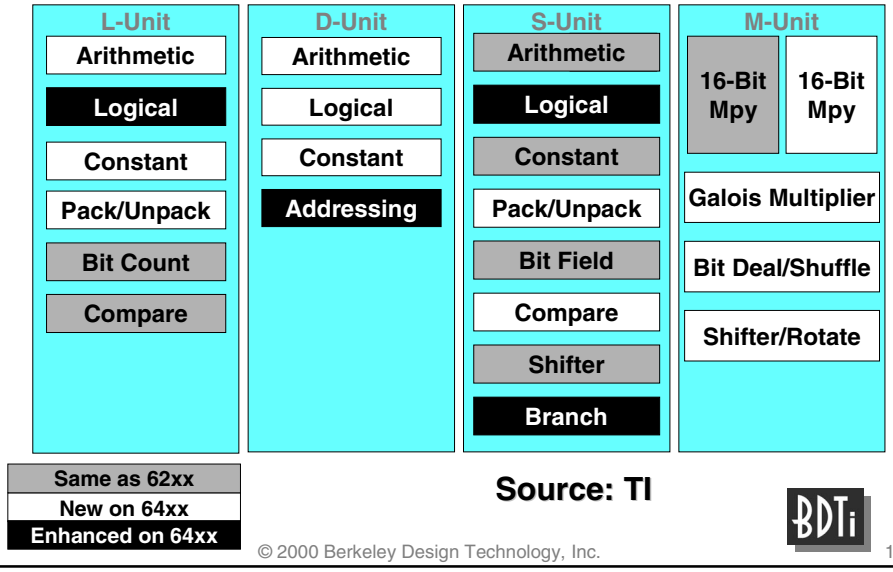
Example CLIW:

```
cliw fft4(r0+=rn0, r1, r4, r5) {
           a2 = a1l * a0h
       || *ma1 = ff1 + ff2
       || *ma2 = a2 - a1h * a0h
       || a1h = *ma3 - *ma4
       || ff1 = *ma3
       || ff2 = *ma4;
}
```

TI TMS320C64xx
'C62xx Gets Major Enhancements

- Still 8-issue architecture, but each instruction and execution unit can do more
- New instructions support SIMD; e.g.,
  - Dual 16-bit multiplies in each multiplier
  - Unaligned loads/stores
  - 8-bit operations for image/video processing
- Data bandwidth doubled: eight 16-bit words/cycle
- Uses dynamic caches
  - Enables high performance without large on-chip RAM
  - A new trend in DSPs, but execution times vary
- Targeting 600 MHz, 1.5 V, samples 1Q01

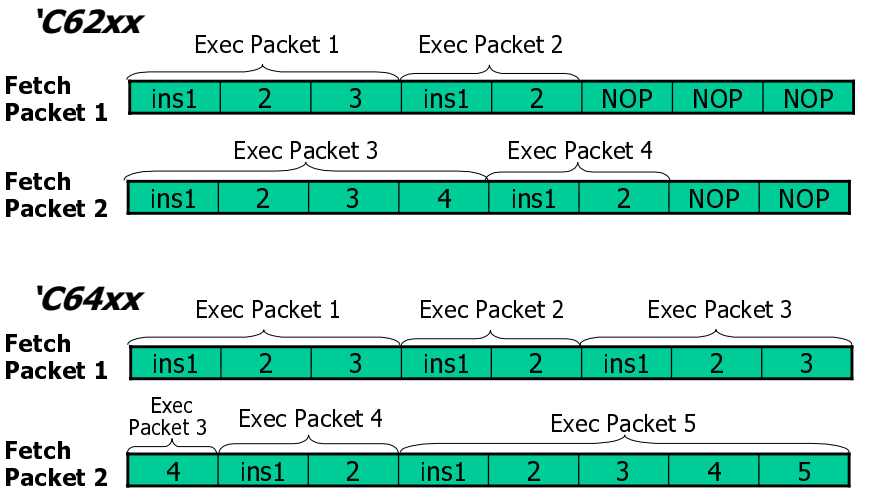
'64xx vs 'C62xx



Source: TI

TI TMS320C64xx

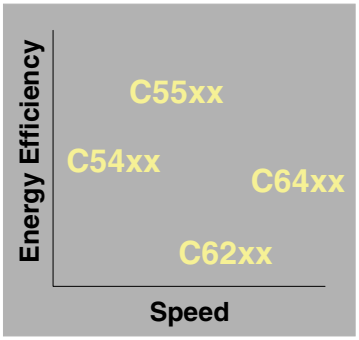
Improves Code Density Over 'C62xx

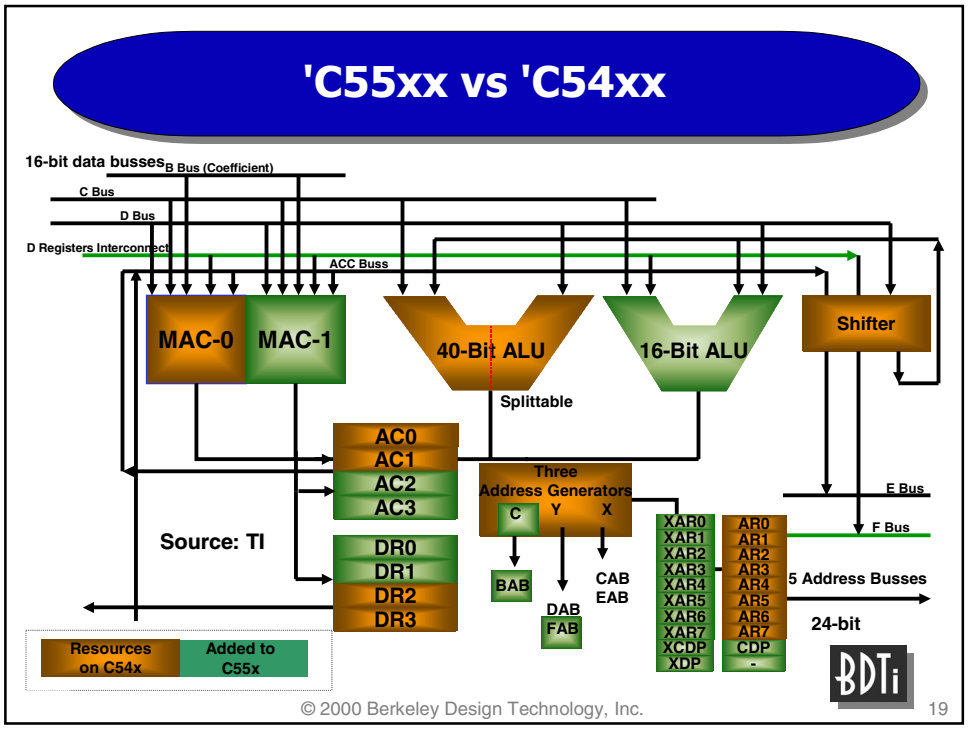


TI TMS320C55xx

The 'C54xx Goes Multi-Issue

- Dual-issue, VLIW (limited)
- Complex, compound instructions
  - Assembly source code compatible with 'C54xx
  - Mixed-width instructions: 8- to 48-bit
- Targeting 160 MHz @ 1.5 V





Summary Comparison
Major VLIW DSP Architectures

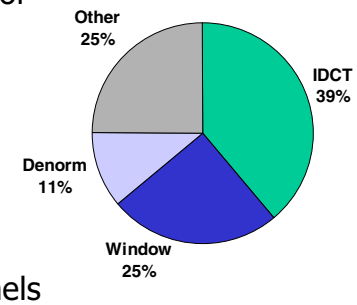
Processor	Issue width	Data memory bandwidth (16-bit words)	Instruction Size	Clock (MHz)	Pipeline Depth	Notable characteristics
TMS320C62xx	8	4 words/cycle	32 bits	300	11	First VLIW-based DSP processor
SC140	6	8 words/cycle	16 bits w/ 16-bit prefixes	300	5	Targeting compact code, low energy
SC110	3	4 words/cycle	16 bits w/ 16-bit prefixes	300*	5	Scaled-down SC140
Carmel	2, 6	4 words/cycle	24/48 bits	180	8	CLIW instructions, 4 AGUs
TMS320C64xx	8	8 words/cycle	32 bits	600*	11	Enhanced 'C62xx
TMS320C55xx	2	3 words/cycle	8-48 bits	160*	7	Based on 'C54xx

*Projected

Algorithm Kernel Benchmarks

A Natural Approach for DSP Benchmarks

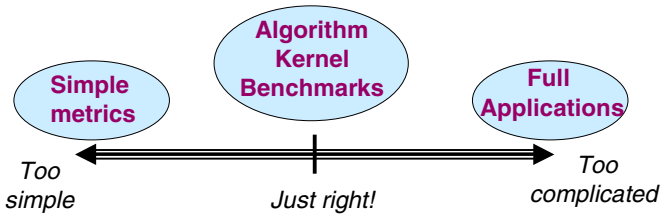
- DSP algorithm kernels are the most computationally intensive portions of DSP applications.
- Example algorithm kernels include
  - FFTs
  - IIR filters
  - Viterbi decoders …
- Application-relevant algorithm kernels are strong predictors of overall performance.

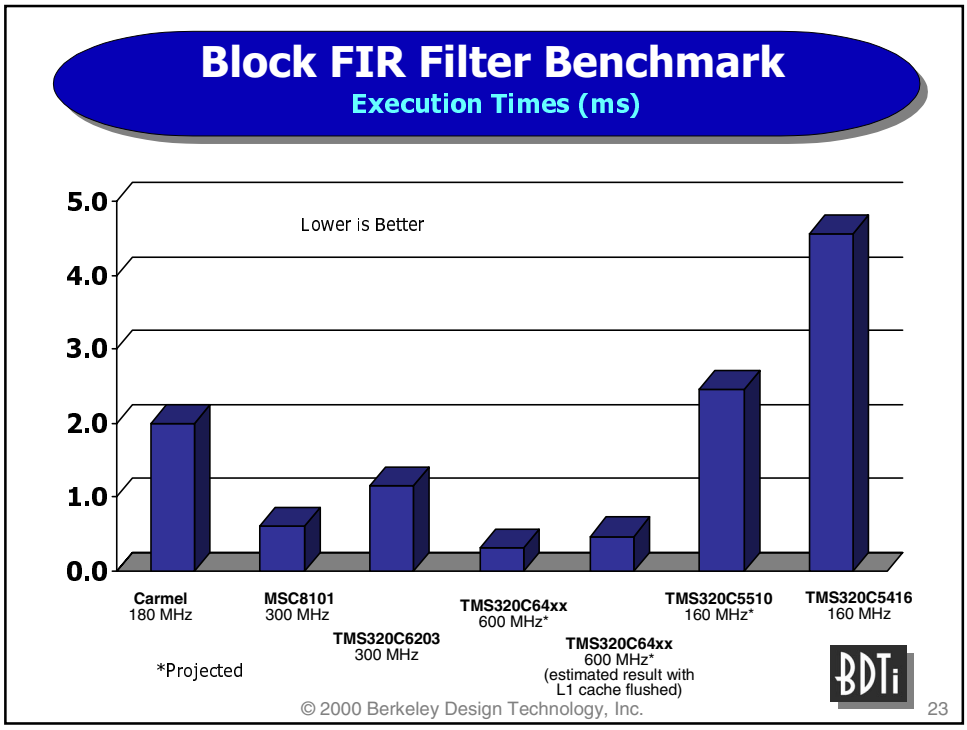


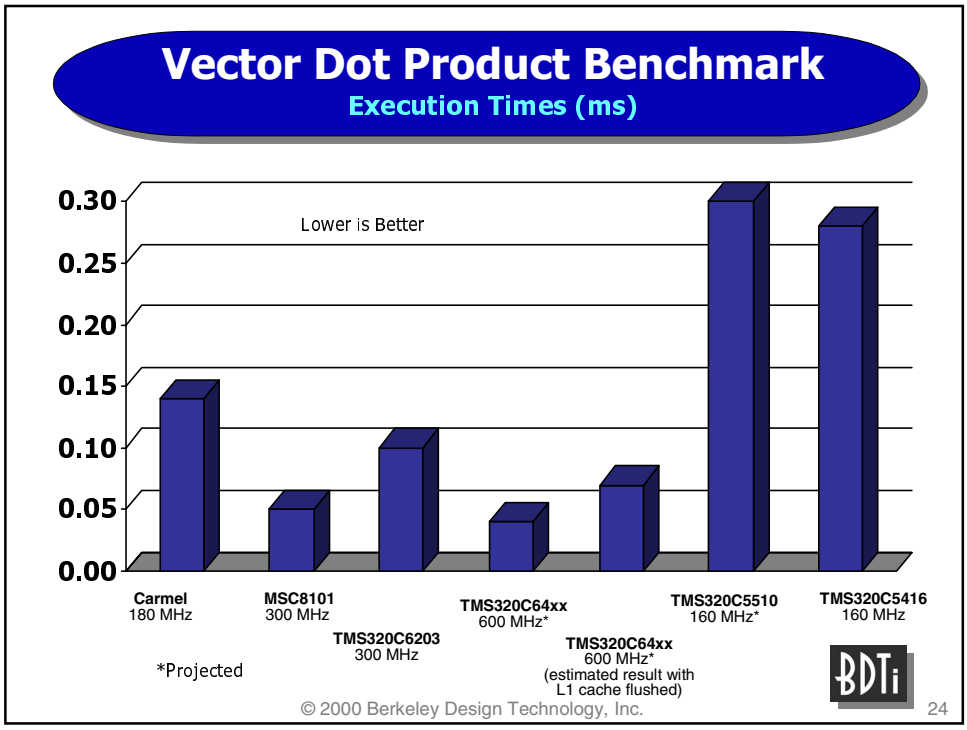
Why Use Algorithm Kernels?

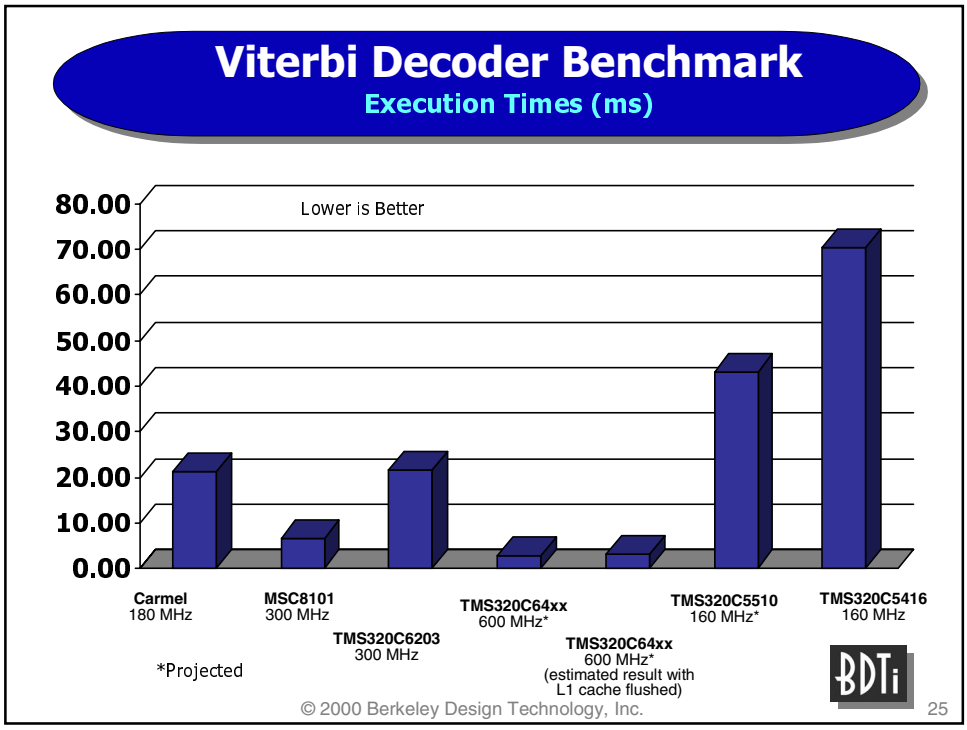
Algorithm kernels are good benchmark candidates because they are:

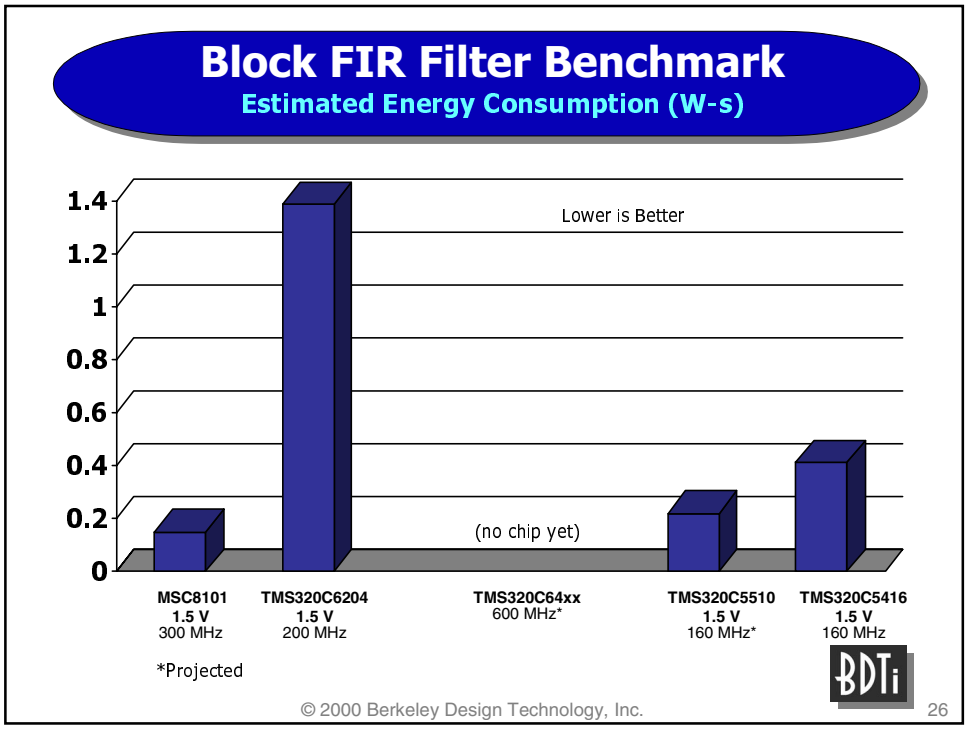
- Relevant
- Practical to specify and implement
- Relatively simple to optimize



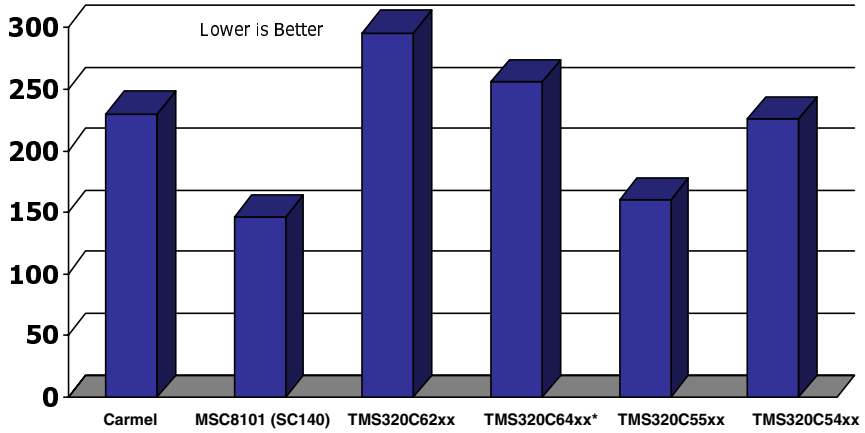






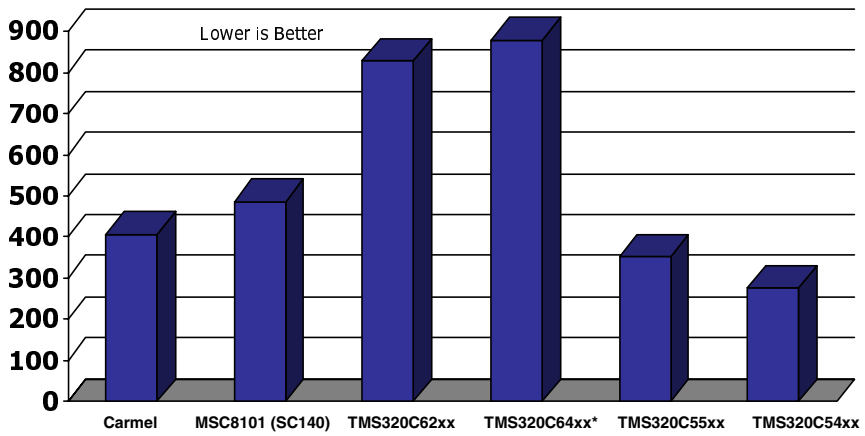


Control Benchmark
Total Memory Usage (Bytes)

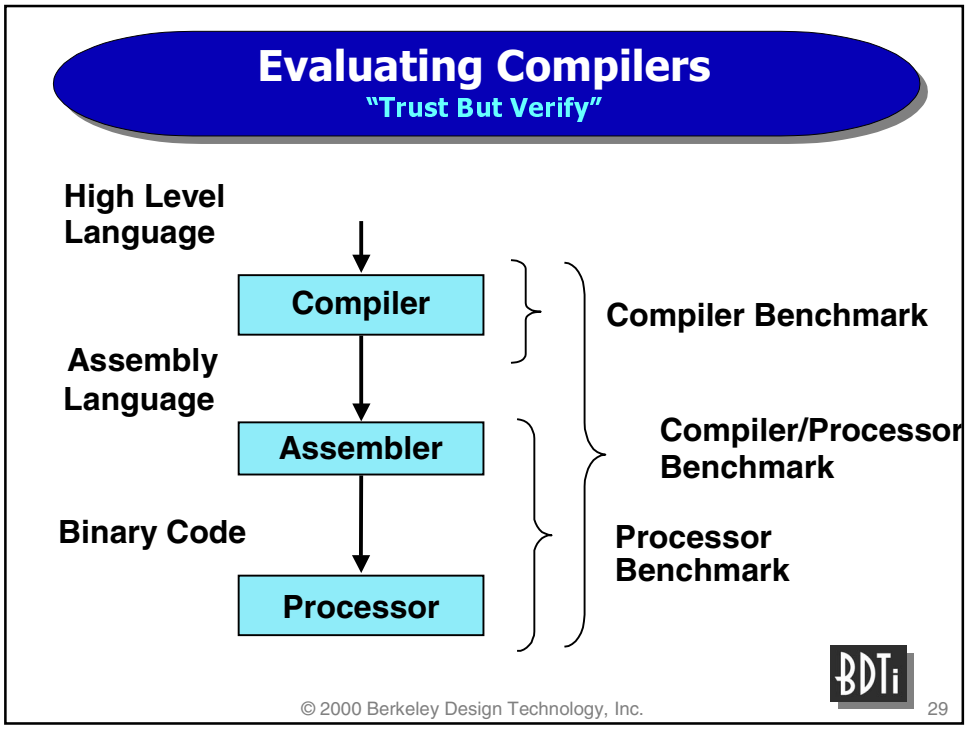


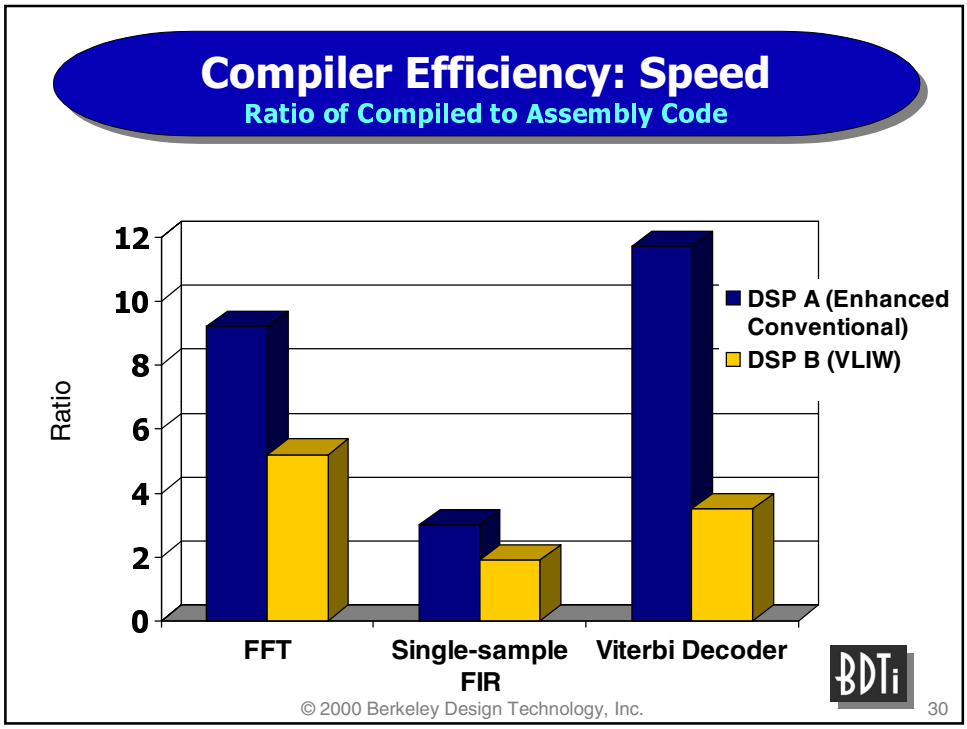
*Projected

Block FIR Filter Benchmark
Total Memory Usage (Bytes)



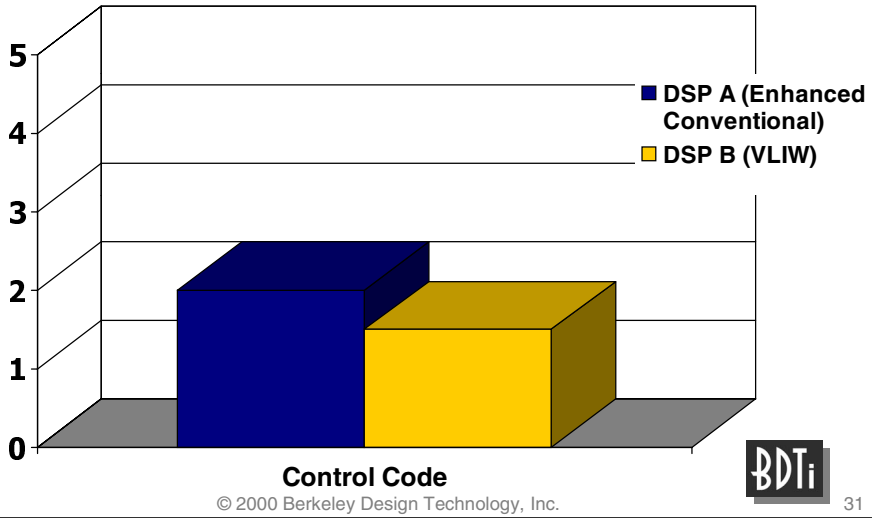
*Projected





Compiler Efficiency: Size

Ratio of Compiled to Assembly Code Size



Conclusions

- Performance…
  - depends on the application
    => Use appropriate benchmarks
  - is more than speed
- VLIW-based DSPs…
  - are speedy
  - *can* be energy efficient or energy hogs
  - *can* be memory efficient or memory hogs
  - *can* be better compiler targets
    - But compilers are far from perfect today
  - are usually not compatible with their predecessors

For More Information...

Free resources on BDTI's web site,

http://www.BDTI.com

- *DSP Processors Hit the Mainstream* covers DSP architectural basics and new developments. Originally published in *IEEE Computer* magazine.
- *Evaluating DSP Processor Performance,* a white paper from BDTI.
- Numerous other BDTI article reprints, slides
- *comp.dsp* FAQ
- *DSP*