Evaluating the Latest DSPs for Portable Communications Applications

Presented by
Jeff Bier
Berkeley Design Technology, Inc. (BDTI)

April 27, 2001

bier@BDTI.com
www.BDTI.com
© 2001 BDTI

Presentation Goals

• Compare and contrast the newest DSP processor architectures
  ■ With a focus on the major vendors
• From an independent, technically based viewpoint:
  ■ Gain an overview of the leading competitor architectures
  ■ Identify key differentiating features, strengths, and weaknesses
  ■ Summarize performance
About BDTI
Independent DSP Analysis • Optimized DSP Software

• Analytical consulting services
• Publications on DSP technology
  ♦ Buyer's Guide to DSP Processors
  ♦ Inside the StarCore SC140
  ♦ DSP Processor Fundamentals
• Training
• Software development services
  ♦ Streaming media applications focus

Low-Power Fixed- Pt. DSPs
Typical applications
  ■ Cellular phones, wireless modems
  ■ Portable audio equipment
  ■ Wearable medical appliances

Key criteria
  ■ Energy efficiency
  ■ Sufficient speed
  ■ Cost
  ■ Memory use
  ■ Small-system integration support
  ■ Tools
  ■ Application-development infrastructure
  ■ Packaging
  ■ Chip-product roadmap
Evaluating the Latest DSPs for Communications Applications

**TMS320C55xx**

- Dual-issue, VLIW (limited)
- Complex, compound instructions
  - Assembly source code compatible with 'C54xx
  - Mixed-width instructions: 8- to 48-bit
- In silicon at 160 MHz @ 1.5 V, ~105 mW
TMS320C55xx
Strengths and Weaknesses

↑ Compatible w/ 'C54xx, and faster
  ↓ Reoptimization often required
  ↓ Performance boost, energy reduction not as dramatic as might be expected
  ↓ 'C54xx code reassembled for 'C55xx may run more slowly at same clock speed

↑ Good code density
  ↓ But not quite as good as SC140

↑ Ample 3rd-party support from the outset
↑ A "safe" choice

TMS320C55xx
Strengths and Weaknesses

↓ Convoluted architecture
↓ Poor compiler target (like most conventional DSPs)
↓ Incompatible with TI's high-performance architectures
↓ Lack of multimedia data type support
↑ Market momentum
  ■ Tools are full-featured, but buggy
  ■ OMAP
**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 SC110 Core**  
Reduced-Cost SC100 Core

- 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
SC110/SC140
Strengths and Weaknesses in Segment

↑ Good technology: strong performance (or potential) on most key metrics (energy, memory, speed)

↓ No terminal-oriented products yet; hence:
   ♦ Integration options unknown
   ♦ Cost unknown
   ♦ Packaging unknown

↑ Strong market positions, credibility of Motorola and Agere in communications

↑ Compatibility with high-performance products

↓ Incompatibility with earlier offerings

SC110/SC140
Strengths and Weaknesses

↓ Lack of multimedia data type support

↓ Initial application development infrastructure limited

↓ Product roadmap unclear (at least to us)
   ♦ E.g., SC140-based parts for terminals?

↑ Multi-vendor, shared architecture ... to some extent

■ Initial tools from StarCore are fairly solid, but unsophisticated
   ■ Little info available on IDEs
Infineon Carmel

- 16-bit fixed-point VLIW DSP core
  - In silicon at 250 MHz, 0.18 μm
- Two data paths, six execution units
  
<table>
<thead>
<tr>
<th>ALU</th>
<th>MAC</th>
<th>EXP</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>MAC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 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"

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 - alh * a0h
  | alh = *ma3 - *ma4
  | ff1 = *ma3
  | ff2 = *ma4;
}
Carmel2000
Customization Via HW Accelerator Blocks

- "PowerPlugs" allow customization of Carmel for target apps; e.g.,
  - MAC PowerPlug
  - MPEG PowerPlug
- Up to 4 PowerPlugs allowed
- Each PowerPlug accessed using one of the slots in a CLIW instruction
  - Operands passed to PowerPlugs in the same manner as to other execution units

Carmel
Strengths and Weaknesses

↑ Fast
  - Not as fast as SC140; faster than SC110
↑ CLIW provides parallelism where needed without bloating code size
  - Good code density (comparable to 'C54xx)
↓ Lack of multimedia data type support
↑ Available as a licensable core
↓ Compiler doesn't support CLIW
↓ Not currently available as a chip
↓ Infineon may be competing with its own Carmel licensees
**Benchmarks: Overall Speed**
(Relative, higher is better)

- 'C54xx, 160 MHz
- 'C55xx, 160 MHz
- SC140, 300 MHz
- SC110, 300 MHz
- Carmel, 250 MHz
- '5685x, 120 MHz
- LSI4xxZ, 200 MHz
- 'C64xx, 600 MHz
- 'C64xx-C, 600 MHz

*E-Estimated  P-Projected*

---

**Viterbi Speed**
(Relative, higher is faster)

- 'C54xx, 160 MHz
- 'C55xx, 160 MHz
- SC140, 300 MHz
- SC110, 300 MHz
- Carmel, 250 MHz
- '5685x, 120 MHz
- 'C64xx, 600 MHz
- 'C64xx-C, 600 MHz

*E-Estimated  P-Projected*
**Benchmarks: Speed/$**

(Relative, higher is better)

<table>
<thead>
<tr>
<th>Device</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>'C54xx, 160 MHz</td>
<td></td>
</tr>
<tr>
<td>'C55xx, 160 MHz</td>
<td></td>
</tr>
<tr>
<td>MSC8101, 300 MHz</td>
<td></td>
</tr>
<tr>
<td>'5685x, 120 MHz</td>
<td></td>
</tr>
<tr>
<td>LSI4xxZ, 200 MHz</td>
<td></td>
</tr>
<tr>
<td>'C64xx-C, 400 MHz</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Estimated</th>
<th>Projected</th>
</tr>
</thead>
<tbody>
<tr>
<td>'C64xx, 400 MHz</td>
<td>[E,P]</td>
</tr>
</tbody>
</table>

**Benchmarks: Energy**

(Relative, lower is better)

<table>
<thead>
<tr>
<th>Device</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>'C54xx, 160 MHz</td>
<td></td>
</tr>
<tr>
<td>'C55xx, 160 MHz</td>
<td></td>
</tr>
<tr>
<td>SC140, 300 MHz</td>
<td></td>
</tr>
<tr>
<td>SC110, 300 MHz</td>
<td>[E,P]</td>
</tr>
<tr>
<td>'5685x, 120 MHz</td>
<td></td>
</tr>
<tr>
<td>LSI4xxZ, 200 MHz</td>
<td></td>
</tr>
<tr>
<td>'C64xx, 600 MHz</td>
<td></td>
</tr>
<tr>
<td>'C64xx-C, 600 MHz</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Estimated</th>
<th>Projected</th>
</tr>
</thead>
<tbody>
<tr>
<td>'C64xx, 600 MHz</td>
<td>[E]</td>
</tr>
</tbody>
</table>
Control Code Memory Use
(Relative, lower is better)

- 'C54xx
- 'C55xx
- SC140
- Carmel
- '5685x
- LSI4xxZ
- 'C62xx
- 'C64xx

E-Estimated
P-Projected

 DSP Software Development
Assembly vs HLLs

- HLLs are irresistible for several reasons
  - Maintainability, productivity, portability
- But C compilers are doubly challenged for DSP
  - Poor semantic match with DSP algorithms
  - DSP architectures are difficult targets
- Performance-critical DSP software often written, optimized in assembly
Compilers
Increasingly Important for Success

- Sophisticated compilers are becoming increasingly vital
  - DSP software is becoming larger and more complex
  - Many new developers tasked with developing DSP software
  - Processor vendors toss compatibility to the wind
  - Processor architectures are becoming more complicated
  - Optimization is often essential

C Language Extensions
Adapting to DSP

- To create usable compilers, language extensions are essential
  - Bridge the semantic gap; e.g.,
    - Fractional data types, intrinsics
  - Bridge the architectural gap; e.g.,
    - Multiple memory spaces
    - Modulo addressing

- Unfortunately,
  - Every vendor has its own extensions
  - But standardization is underway
Evaluating Compilers
“Trust But Verify”

High Level Language → Compiler → Compiler Benchmark
Assembly Language → Assembler
Binary Code → Processor

Compiler Benchmark
Compiler/Processor Benchmark
Processor Benchmark

Compiler Efficiency: Speed
Ratio of Compiled to Assembly Code

Ratio
FFT  Single-sample FIR  Viterbi Decoder

DSP A (Enhanced Conventional)
DSP B (VLIW)

© 2001 BDTI
Compilers
Look Before You Leap

- Robustness
- Efficiency
  - Speed of generated code
  - Size of generated code
  - Ease of obtaining efficient generated code
  - Ability to control optimizations
- Language extensions
  - Beware of “check the box” syndrome
- Debugging, profiling capabilities
- Documentation, support
Alternatives

A broad spectrum, with overlapping regions:

- **DSPs**
  - TMS320C54xx, TMS320C55xx, Frio, Carmel2000
- **Cores**
  - Teak, Palm, 3DSP, LSI40xZ
- **Customizable cores**
  - Tensilica Xtensa, ARC, Improv
- **Reconfigurable processors**
  - Morphics
- **ASSPs**
  - Conexant, LSI Logic, Infineon
- **GPPs and hybrids**
  - SH3-DSP, ARM9E

Which Architecture(s) Will Dominate DSP?

- A new wave of architectural diversification and specialization; e.g.,
  - Mixing of DSP and GPP family trees
  - VLIW DSPs
  - User-defined instructions
  - More application-specific hardware
  - Reconfigurable processors
- There is no overpowering need to standardize on an architecture
- Different architectures will succeed in different applications
Industry Trends
New Competition

Competition is increasing, and increasingly vigorous, because:

- A large, profitable, growing market attracts new entrants, large and small
  - E.g., Intel, 3DSP
- GPP/DSP convergence
- ASSP/DSP overlap
- New technologies pursued by new players

Industry Trends
Critical Challenges

- How wide a range of applications can be well served by a single architecture?
- Increase DSP performance while reducing cost, energy, memory use
  - And easing development
- Software development
  - Tools, especially compilers
- Integration, system-on-chip design
  - Increasing application content
Conclusions
Comparing Performance

- Performance is more than speed
  - Cost/perf, energy efficiency, memory use, …
- Vendor performance claims should be viewed skeptically
  - “MIPS” = …
- Benchmarks are invaluable, but
  - Use appropriate benchmarks
  - Benchmarks are a sharp tool
- Beware of vaporware!

Conclusions
Comparing Processors

- Performance is interesting, but other factors are equally important
- Architecture is interesting, but it isn’t all-important
- Compare processors the way you’d compare cars
  - Not exclusively on their top speed
  - Suitability for the task at hand
  - “Cost of ownership”
  - Time to market, ease of use, …
Resources from BDTI

- www.BDTI.com
  - Pocket Guide to DSP Processors
  - BDTImark2000™ DSP benchmark scores
  - White papers, article reprints
- Buyer's Guide to DSP Processors
  - Includes the SC140, MSC8101, TMS320C64xx, TMS320C55xx, TigerSHARC, and many others
- Inside the StarCore SC140
  - Includes the SC140 and MSC8101
- Forthcoming Inside reports:
  - SC110, Frio, LSI4xxZ, ...
Appendix: Other Processors

TMS320C62xx
The Original VLIW DSP Processor

2 independent data paths, 8 execution units

On-Chip Program Memory

Dispatch Unit

32x8=256 bits (8 instructions)

L1 S1 M1 D1
Register File A

L2 S2 M2 D2
Register File B

On-Chip Data Memory

L=40-bit ALU
S=32-bit ALU, 40-bit Shifter
M=Multiplier
D=32-bit Add/Sub for Address Generation
TMS320C64xx

- 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
  - 8-bit operations for image/video processing
  - Unaligned loads/stores
- 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

TMS320C64xx

- Compatible: runs 'C62xx object code
- Targeting 600 MHz, 1.5 V, samples June ‘01
- Three family members announced
  - TMS320C6414: General-purpose
    - 64-channel DMA, 2 EMIs, 3 serial ports, 32-bit host port
  - TMS320C6415: Networking
    - PCI, ATM interfaces
  - TMS320C6416: 3G wireless infrastructure
    - Viterbi coprocessor, "Turbo" coprocessor, PCI, ATM
Evaluating the Latest DSPs for Communications Applications

TMS320C64xx

<table>
<thead>
<tr>
<th>L-Unit</th>
<th>D-Unit</th>
<th>S-Unit</th>
<th>M-Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic</td>
<td>Arithmetic</td>
<td>Arithmetic</td>
<td>16-Bit Mpy</td>
</tr>
<tr>
<td>Logical</td>
<td>Logical</td>
<td>Logical</td>
<td>16-Bit Mpy</td>
</tr>
<tr>
<td>Constant</td>
<td>Constant</td>
<td>Constant</td>
<td>Galois Multiplier</td>
</tr>
<tr>
<td>Pack/Unpack</td>
<td>Addressing</td>
<td>Pack/Unpack</td>
<td>Bit Deal/Shuffle</td>
</tr>
<tr>
<td>Bit Count</td>
<td></td>
<td>Bit Field</td>
<td>Shifter/Rotate</td>
</tr>
<tr>
<td>Compare</td>
<td></td>
<td>Compare</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shifter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Branch</td>
<td></td>
</tr>
</tbody>
</table>

- Same as 62xx
- New on 64xx
- Enhanced on 64xx

Source: TI

TMS320C64xx

Improved Code Density

'C62xx

Fetch Packet 1

<table>
<thead>
<tr>
<th>Exec Packet 1</th>
<th>Exec Packet 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ins1 2 3</td>
<td>ins1 2 NOP NOP NOP</td>
</tr>
</tbody>
</table>

Fetch Packet 2

<table>
<thead>
<tr>
<th>Exec Packet 3</th>
<th>Exec Packet 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ins1 2 3 4</td>
<td>ins1 2 NOP NOP</td>
</tr>
</tbody>
</table>

'C64xx

Fetch Packet 1

<table>
<thead>
<tr>
<th>Exec Packet 1</th>
<th>Exec Packet 2</th>
<th>Exec Packet 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ins1 2 3</td>
<td>ins1 2 ins1 2 3</td>
<td></td>
</tr>
</tbody>
</table>

Fetch Packet 2

<table>
<thead>
<tr>
<th>Exec Packet 3</th>
<th>Exec Packet 4</th>
<th>Exec Packet 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 ins1 2</td>
<td>ins1 2 3 4</td>
<td>5</td>
</tr>
</tbody>
</table>

© 2001 BDTI
TMS320C64xx
Strengths and Weaknesses

↑ Very fast, particularly on imaging and SIMD-friendly algorithms

↑ Compatibility
  ↑ With 'C62xx provides upgrade path
  ↑ With 'C67xx provides upgrade path, prototyping options

↑ Multimedia data type support
↑ Builds on maturity of 'C62xx tools
↑ More complex instructions (e.g., dot product instruction) improve code density, but...

↓ More complex instructions make the compiler’s job harder
  ↓ And 'C6xxx processors are an assembly programmer’s worst nightmare

↓ Many of the same problems as 'C62xx
  ↓ Not interruptible in tight loops
  ↓ Deep, complex pipeline
  ↓ Poor code density

↓ Caches reduce execution-time predictability
↓ Simulator not currently cycle-accurate
↓ Incompatible with TI's low-power procs
**LSI Logic LSI40xZ**

- 4-way superscalar machine
- 16-bit fixed-point DSP
  - Dual-MAC unit, 2 ALUs, shifter
- Some SIMD capabilities
- Borrows and adapts features from high-performance GPPs
  - Run-time instruction scheduling
  - Branch prediction
  - RISC-like load/store instruction set
  - Instruction and data caches

**LSI40xZ**

- LSI Logic is using the core for:
  - ASICs
  - ASSPs
  - General-purpose DSPs
  - Licensing
    - IBM, Broadcom, Conexant have licensed
- Emphasizing communications infrastructure applications
- At 200 MHz, a moderately high-performance DSP
  - Leading speed in 1998; less so today
## LSI40xZ

**Strengths and Weaknesses**

- Available in core *and* chip form
- Very good code density
- Architecture has support of multiple chip vendors: LSI, Broadcom, IBM, Conexant
- Speed doesn't approach that of SC140 or 'C64xx
- Superscalar approach detracts from execution-time predictability

**Quality of tools?**
- Simulator not quite cycle-accurate (±5%, according to LSI)

## ADI/Intel Frio

**Data Path**

- 2 16 x 16 MACs
- 2 40-bit ALUs
- 8 Registers
- Barrel Shifter
Frio

- Single-issue mixed-width instruction set (16/32/64)
  - Not a VLIW architecture
  - "RISC" instructions
- ADI/Intel have been very circumspect; limited info available
- Architecture will be scaled, according to ADI/Intel
- Targeting 300 MHz for initial samples

Frio

- ADI/Intel emphasizing low power, compilability
  - "Dynamic power management"
  - Targeting apps requiring control + DSP
- Likely to be between the SC140 and SC110 in terms of speed
- Partnership may face similar challenges to StarCore
Frio
Strengths and Weaknesses

▼ Slow to the starting gate
▼ ADI architecture proliferation
▼ Incompatible with earlier architectures
▼ Incompatible with rest of partners’ processors
▲ Multi-vendor, shared architecture... to some extent