Why not just use a RISC chip?

What is RISC?
“Any microprocessor architecture designed after 1986.”

- Load/store architecture
- 32, 64b IEEE fp add, multiply, divide, sqrt
- Elaborate exception handling
- MMU for memory hierarchy
- TLB for virtual memory (also protection)
- “Low power” means about 2 watts (vs. 10)
Why use Programmable DSPs?

What is a programmable DSP?
“Any microcomputer still programmed in assembly language”

- Can be small, cheap.
- “Marching band” timing.
- Little or no memory hierarchy.
- Low power means 50mW.
- Low system cost.

Fixed-point vs floating-point

“Generation gap”

**Fixed-point**
- Accumulator architecture
- Fewer registers
- 16 or 24-bit integer data
- Cheaper

**Floating-point**
- Load/store or memory-register architecture
- More registers
- 32-bit integer or floating point (IEEE or native) data
- More expensive
- Good compilers (C, C++, Ada)
RISC vs DSP: Instruction encoding

**RISC: Super-scalar**

- Reorder
- Load/store
- FP Unit
- Integer Unit

**DSP: Horizontal microcode**

- Load/store
- Load/store
- Address
- Multiplier
- Address

RISC vs DSP: Memory hierarchy

**RISC**

- Registers
- Out-of-order
- I/D-Cache
- Physical memory
- Virtual memory
- Register window

**DSP**

- Registers
- I-Cache
- Internal memories
- External memories
- DMA Controller
RISC vs DSP: Performance and cost

<table>
<thead>
<tr>
<th></th>
<th>RISC</th>
<th>DSP</th>
<th>Parallel DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signal processing</strong></td>
<td>![Graph]</td>
<td>![Graph]</td>
<td>![Graph]</td>
</tr>
<tr>
<td><strong>Symbolic processing</strong></td>
<td>![Graph]</td>
<td>![Graph]</td>
<td>![Graph]</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>![Graph]</td>
<td>![Graph]</td>
<td>![Graph]</td>
</tr>
</tbody>
</table>

Addressing Modes

<table>
<thead>
<tr>
<th>Addressing Mode</th>
<th>TMS320C50</th>
<th>TMS320C30</th>
</tr>
</thead>
<tbody>
<tr>
<td>immediate</td>
<td>ADD #0FFh</td>
<td>addi 32,r0</td>
</tr>
<tr>
<td>register</td>
<td>(implied)</td>
<td>addi r1,r0</td>
</tr>
<tr>
<td>direct</td>
<td>ADD 010h</td>
<td>addi @count,r0</td>
</tr>
<tr>
<td>indirect</td>
<td>ADD *</td>
<td>addi *+ar2(7),r0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>addi *ar2++,r0</td>
</tr>
</tbody>
</table>
Fixed-point arithmetic

Decimal:

\[
\begin{array}{c}
99.999 + 9.99 = 109.989 \\
99.999 \times 9.99 = 998.99001
\end{array}
\]

Binary:

\[
\begin{array}{c}
0.111 + 0.111 = 1.110 \\
0.111 \times 0.111 = 0.0110001
\end{array}
\]

\[
0.111 \quad 7/8
\]

\[
\begin{array}{c}
0.111 \times 1.000 = 1.111000 \\
= 11.111000 \quad 56/64
\end{array}
\]

Except for one special case, the sign and integer bit are always the same, so a fix-point multiply means:

- Multiply two N-bit words
- Shift 2N-bit product left one bit
- Take high-order N bits

Some examples of products with prog. DSPs

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
<th>Description</th>
<th>DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digicom</td>
<td>Various Modems</td>
<td>V.32, V.32bis, V.22bis, Fax</td>
<td>ADSP-21xx</td>
</tr>
<tr>
<td>Ericsson</td>
<td>Hotline GH197</td>
<td>GSM digital cellular phone</td>
<td>ADSP-2102</td>
</tr>
<tr>
<td>E-mu</td>
<td>Proteus MPS</td>
<td>Music effects</td>
<td>ADSP-2105</td>
</tr>
<tr>
<td>Intel</td>
<td>ActionMedia II</td>
<td>DVI product series</td>
<td>ADSP-2105</td>
</tr>
<tr>
<td>Peavy Electr.</td>
<td>Spectrum Bass</td>
<td>Sound synthesis</td>
<td>ADSP-2105</td>
</tr>
<tr>
<td></td>
<td>Tone Module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peavy Electr.</td>
<td>IDL 1000</td>
<td>Digital delay for audio</td>
<td>ADSP-2105</td>
</tr>
<tr>
<td>Sharp</td>
<td>JY-7500 MOD Drive</td>
<td>Magneto optical disk drive (servo loop)</td>
<td>ADSP-2101</td>
</tr>
<tr>
<td>Siemens</td>
<td>NNSR</td>
<td>Neural net speech recognizer</td>
<td>ADSP-2111</td>
</tr>
<tr>
<td>Xing</td>
<td>VT-Compress</td>
<td>JPEG/MPEG image compression</td>
<td>ADSP-2105</td>
</tr>
</tbody>
</table>