Native Signal Processing With Altivec in the Ptolemy Environment
Ken Aponte & Ken Logan
EE382C Embedded Software Systems
Spring 2000
## NSP Extensions

(Approx. Chronologically Ordered)

<table>
<thead>
<tr>
<th>Architecture</th>
<th>NSP Extension (# Instructions)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun UltraSPARC</td>
<td>VIS (50+)</td>
<td>No Saturation Arithmetic, FP Registers used for vector registers</td>
</tr>
<tr>
<td>HP PA-RISC</td>
<td>MAX-1 (?)</td>
<td>32 bit wide (MAX-1)</td>
</tr>
<tr>
<td></td>
<td>MAX-2 (?)</td>
<td>64 bit wide (MAX-2)</td>
</tr>
<tr>
<td>SGI MIPS</td>
<td>MDMX (?)</td>
<td>32 64 bit FP regs. (+192 bit accumulator)</td>
</tr>
<tr>
<td>Digital Alpha</td>
<td>MVI (13)</td>
<td></td>
</tr>
<tr>
<td>Intel x86</td>
<td>MMX (57)</td>
<td>8 FP Registers used for MMX regs</td>
</tr>
<tr>
<td></td>
<td>SSE (71)</td>
<td>8 dedicated SSE reg</td>
</tr>
<tr>
<td>Amd x86</td>
<td>3DNow! (21)</td>
<td>Advertise improvements on MMX context switch time.</td>
</tr>
<tr>
<td></td>
<td>Enhanced 3DNow! (45)</td>
<td></td>
</tr>
<tr>
<td>PowerPC</td>
<td>Altivec (162)</td>
<td>32 128 bit registers added</td>
</tr>
</tbody>
</table>
NSP Extensions Use SIMD Semantics

SIMD benefits:
• Takes advantage of data level parallelism to provide order of magnitude speedups for many operations common in signal processing.
• One vector instruction can do work of many scalar instructions, reducing working set size for instructions. (Benefits i-cache, saves memory bandwidth for data).

SIMD drawbacks:
• Auto-vectorizing compilers still in experimental stage of development.
• SIMD does not provide any speedup for some algorithms.
• Existing applications structured for scalar computation semantics.
• Vector data-types not built into ANSI C; C compilers must be extended (hacked) to generate good SIMD code (using inline assembly macros is a common workaround).
• Look and feel of various NSP architectures varies widely.
Code-Generation in Ptolemy

Ptolemy: Simulation and prototyping of heterogeneous systems

- Agility – supports distinct computational models so that each can be simulated in a manner appropriate and natural to that system
- Heterogeneous – allows distinct computational models to co-exist seamlessly for the purpose of studying their interactions
- Extensible – easy integration of new computational models without changes to existing ones
- Ease of Use – Graphical interface at an abstract, readable level

Code-Generation:

- Part of the software synthesis flow: partition -> schedule -> generate
- The ability to target a specific architecture or environment
- Ease of re-targeting, re-partitioning, and re-synthesis through the use of code domains
NSP Applications

• Certain algorithms take advantage of NSP
  – Median Filters
  – FFT, FIR
  – Convolution
  – Matrix math
  – Dequantizer and Filter Banks in MPEG decoders

• Existing stars in Ptolemy targeting NSP
  – UltraSparc VIS (FIR, FFT)
  – Motorola DSP 56000
Plans & Goals

- **Main goal** is to benchmark effectiveness of using NSP extensions without resorting to hand-written assembly code.
- Will implement new Ptolemy stars with ability to generate C code that utilizes Altivec technology.
- Will use the newly released Altivec enabled gcc to compile C code that we will generate in the Ptolemy framework.
- Will use a timing model (available publicly from Apple) to determine performance differences between scalar and vectorized code compiled for the MPC7400 PowerPC processor (G4).
- We will learn about Ptolemy’s code generation abilities, work with some signal processing algorithms, use a cutting-edge NSP extension, and do a performance analysis in the process.