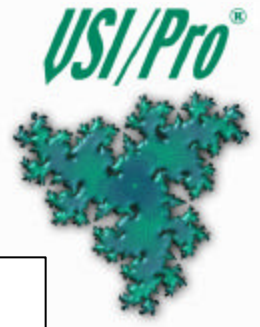
MPI Software Technology, Inc.

VSIPL for Diverse Architectures (Pentium 4 to DSPs)

Anthony Skjellum
Brian Chase
Wenhao Wu

September 23, 2003

Prelude – VSI/Pro



- Overview: Current VSIPL platform support
- Status: G4 / Altivec
- Widely used worldwide
- Domestic production computing adoption picking up
- Helps untie programs from specific vendors
- Expertise on optimizing G4 a major part of expertise
- Expertise on porting to different PPC environments also key expertise
- Dealing with C/C++ toolchains a major expertise
- Key optimizations for more advanced users (e.g., Rader's algorithm and other NTT-motivated improved) with high performance are at cusp of newest release efforts
- Complete version for Image processing also released
- Customers have started asking for non-G4/Altivec alternatives!

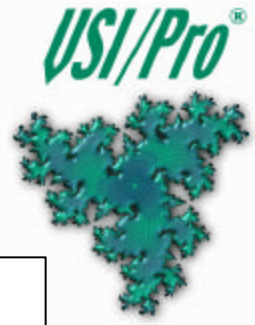
MSTI's Strategy

Availability on Different Processors

- Core+ G4 / Altivec
- Core P4 / SSE
- CoreLite TI DSP C67 family

Operating System / Development Tools

- VxWorks, MercuryOS, LynxOS, Linux, MacOSX
- Windows, Linux, VxWorks
- Code Composer toolset



Why P4?

- The higher clock speed, 3 or more GHz
- COTS technology enables cost effective solutions
- Anticipated lower power versions from Intel and third parties in future
- Not all embedded systems equally power/heat constrained even now
- Double precision 4-way vectorization useful
- Future winner in Gflop/Watt? Gflop/$?

Why C67

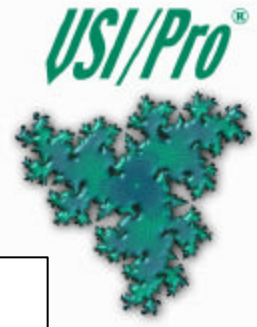
- Specially designed architecture for DSP applications.
  - very deep pipeline
  - very large instruction word (VLIW) architecture
  - streaming data
- Better GFlops per $ than G4 / Altivec

Product Exploration/Results - P4

- Full Core profile support for Windows, Linux, and VxWorks.
- Optimized FFT performance for SSE registers (performance graph later)
- Optimized matrix library easily achievable also
- Can equal or beat MKL (Intel commercial library) in significant aspects of overall performance… more tuning possible

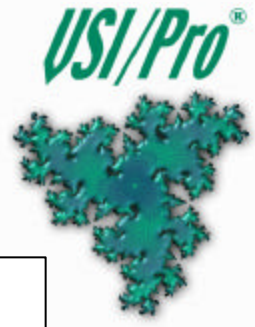
Porting Experience for C67

- What we achieved in 1 month:
- VSI/Pro Core Lite profile is completely ported for TI C67.
- We have C6711 optimized Complex-to-Complex inplace and out of place, forward and inverse FFTs:
  - vsip_ccfftop_f()
  - vsip_ccfftip_f()
  - C6711 150Mhz CPU 29300 cycles for 1024 element FFT



Issues

- C side: Straightforward
- C++ side: Strict on template support
- VLIW assembly side: No hand tuned assembly code in the library yet… next step before product release



C67 Operating Systems

Example Various OS platforms:

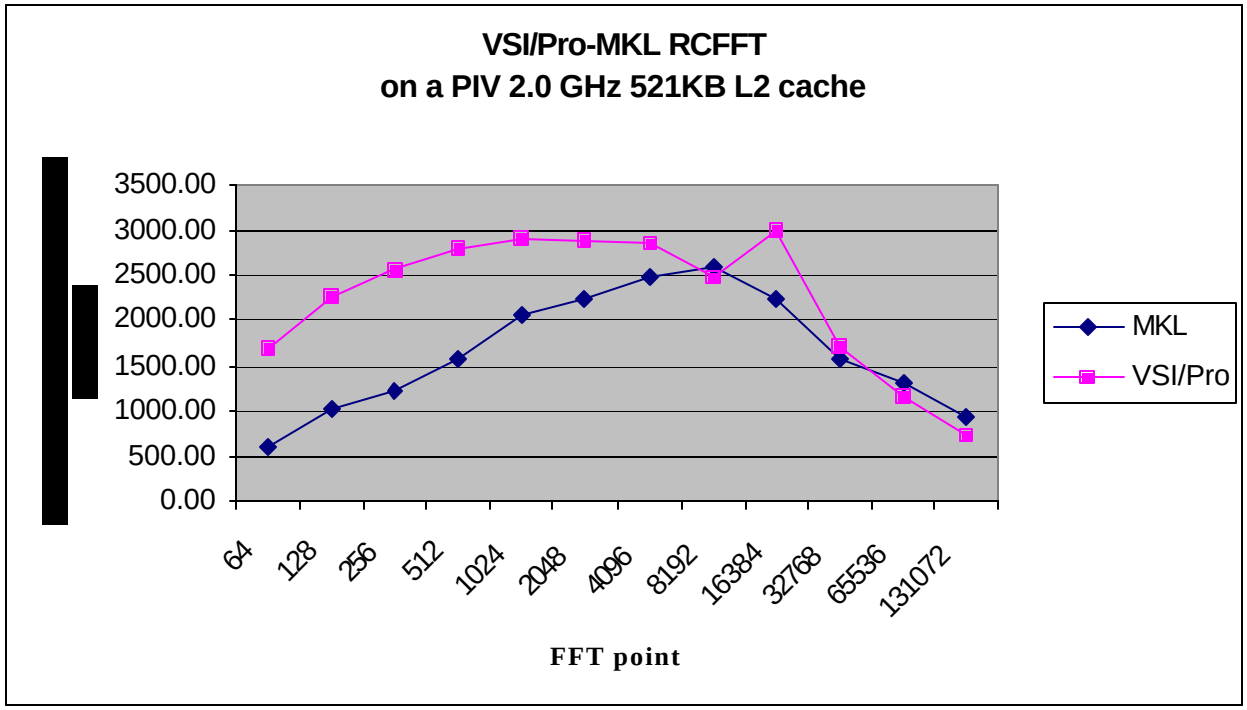
- SPARK (Small Portable Adjustable Real-time Kernel)
- OSE
- Diamond
- Thread XABS GmbH Jena

FFT optimization for P4

- Algorithm was engineered for the architecture to minimize problems arising from the scarcity of registers and lower cache associativity.
- The algorithm is auto-sort DIF, efficient not only on power-of-two sizes.
- The key functions are written in assembler supported by highly optimized C and C++ code, using SSE.

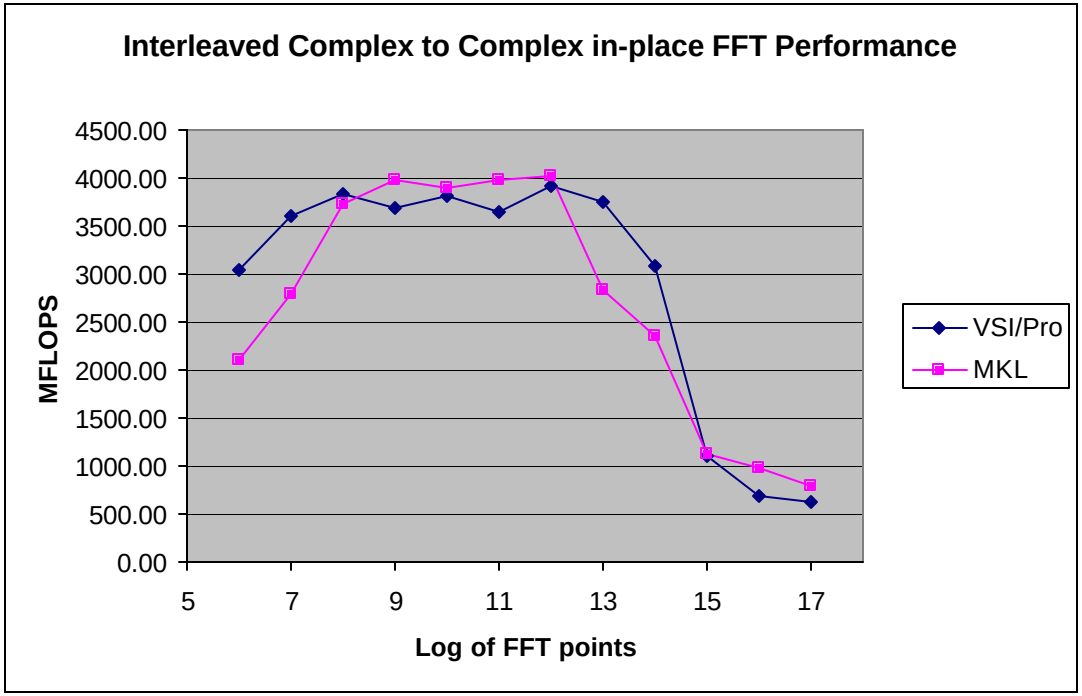
FFT performance on P4

- RCFFT Comparison with MKL…



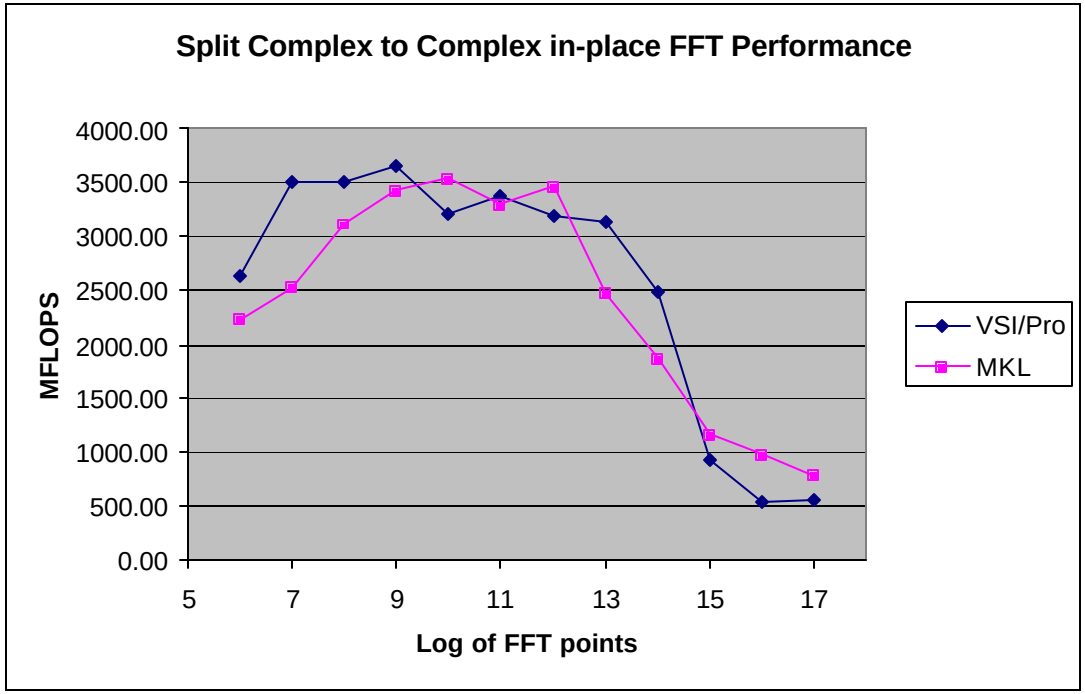
FFT performance on P4

Interleaved in-place CCFFT Comparison



FFT performance on P4

Split in-place CCFFT



FFT optimization for DSP

- Using Radix-2 , Radix-4 algorithms.
- Also using cache splitting (the L1 cache is 4KB, so the splitting is needed for sizes > 256 for in-place FFT) .

PRELIMINARY
FFT performance on C67

TI DSP C6711 150 MHz DSK

	ccfftop fwd	ccfftop inv	ccfftip fwd	ccfftip inv
Length(N)	scale = 1.0	scale = 1/N	scale = 1.0	scale = 1/N
64	204	211	256	239
128	252	262	308	287
256	297	292	302	295
512	259	265	263	266
1024	257	258	259	261
2048	205	201	239	236

Future Plans for C67

- Release the Core Lite profile library for the C67 platform
- Explore releasing Core profile library
- Explore possibilities of partnering with OS vendors such as OSE Systems

P4-related Issues, I

- Single-precision optimization not a big concern outside embedded computing…
- Good free libraries exist (e.g., FFTW, LAPACK) and MKL exists as alternatives
- Academic basic kernels for matrix multiplication (non-ATLAS) are now mature enough to use with small code size, but these are not open source/redistributable (e.g., libgoto)
- Several universities working on better free libraries
- Code bloat an issue for certain library architectures when considering embedded (e.g., ATLAS code size)
- The merger of free libraries and free VSIPL has been tried, not as good a fully optimized library (e.g., VSIPL ERI, VSIPL Ref Implementation upgrade)
- Demand for commercial VSIPL for P4 remains a question

P4-related Issues, II

- Distinct flavors of P4 (e.g., Athlon) have distinctively different optimal libraries
  - Cache architecture
  - Instruction decode differences
  - TLB and other memory issues
  - Register file differences (e.g., 16 vs 8)
- Strong potential that future embedded P4 clones will also have different optimal choices in their hardware configurations

Why we think it is useful to have commercial VSIPL on P4 and C67

- Shows true performance portability story between diverse architectures, not just different G4/Altivec OS's and vendors
- Allows system designers to work with assumption low software porting cost, and explore other aspects of design alternatives
- Processors are getting harder to program
- Precise mix of required optimizations for embedded not strong emphasis of free libraries per se

Conclusions

- Demand for VSIPL for non-G4 platforms is TBD… appears promising but not well developed
- Opportunities to achieve extremely high performance on clearly different architectures now evident
- Proof of concept may help drive adoption
- Technical hurdles involving hand-optimization remain for key inner kernels on each new platform, but do not require massive coding in assembly language if handled correctly
- C/C++ toolchain always an issue for new processor + OS combinations