

Implementing VSIPL Using Intelligent Compiler Technology

Steve Paavola
SKY Computers, Inc.

Abstract

SKY Computers has implemented the entire VSIPL core library (~1500 routines) into their AltiVec based Merlin product family using the SKYvec Vectorizing Compiler technology. The SKYvec intelligent compiler technology enabled SKY to port the library within a very short amount of time with hand coded library performance. The lessons learned as well as performance comparisons between VSIPL and SKY's Standard Math Library will be presented.

Implementing VS IPL Using Intelligent Compiler Technology

Stephen Paavola
Paavola@skycomputers.com

- ❑ **Originally funded by DARPA**
- ❑ **Final specification released March, 2000**
- ❑ **Implementations**
 - Randy Judd - SPAWAR SYSCEN - “Core Plus”
 - CSPI - “Core Lite” conformance
 - Mercury Computers - “Core Lite” conformance
 - MPI Softech - “Core Lite” conformance
 - SKY Computers - “Core Plus”

Conformance

☐ **“Core Lite”**

- **Enough functions to be useful for some applications**
- **Basic vector math, FFT and FIR filters**
- **One integer data type, one floating point data type**

☐ **“Core”**

- **Essentially the entire specification**
- **Vector and Matrix functions, including matrix solvers**
- **One integer data type, one floating point data type**

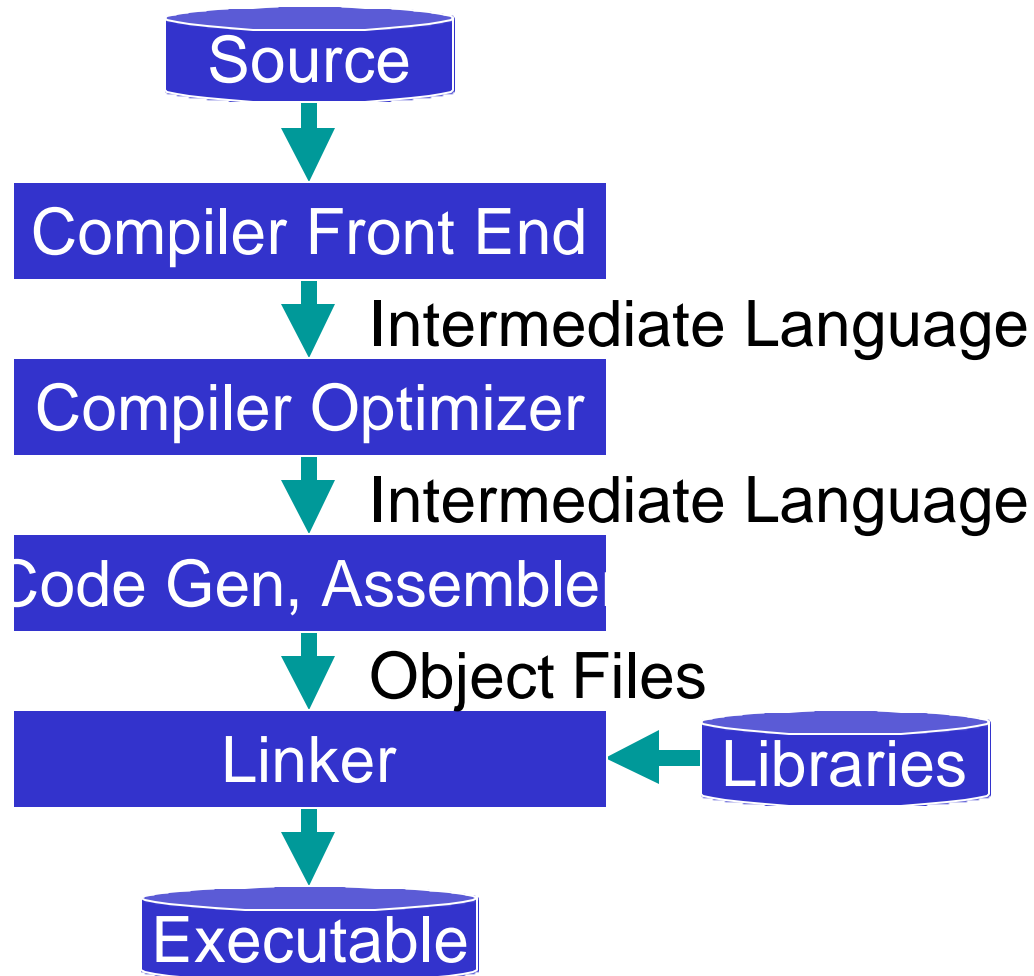
☐ **SKY’s “Core Plus”**

- **Core compliance**
- **Two floating point data types**
- **Six integer data types**

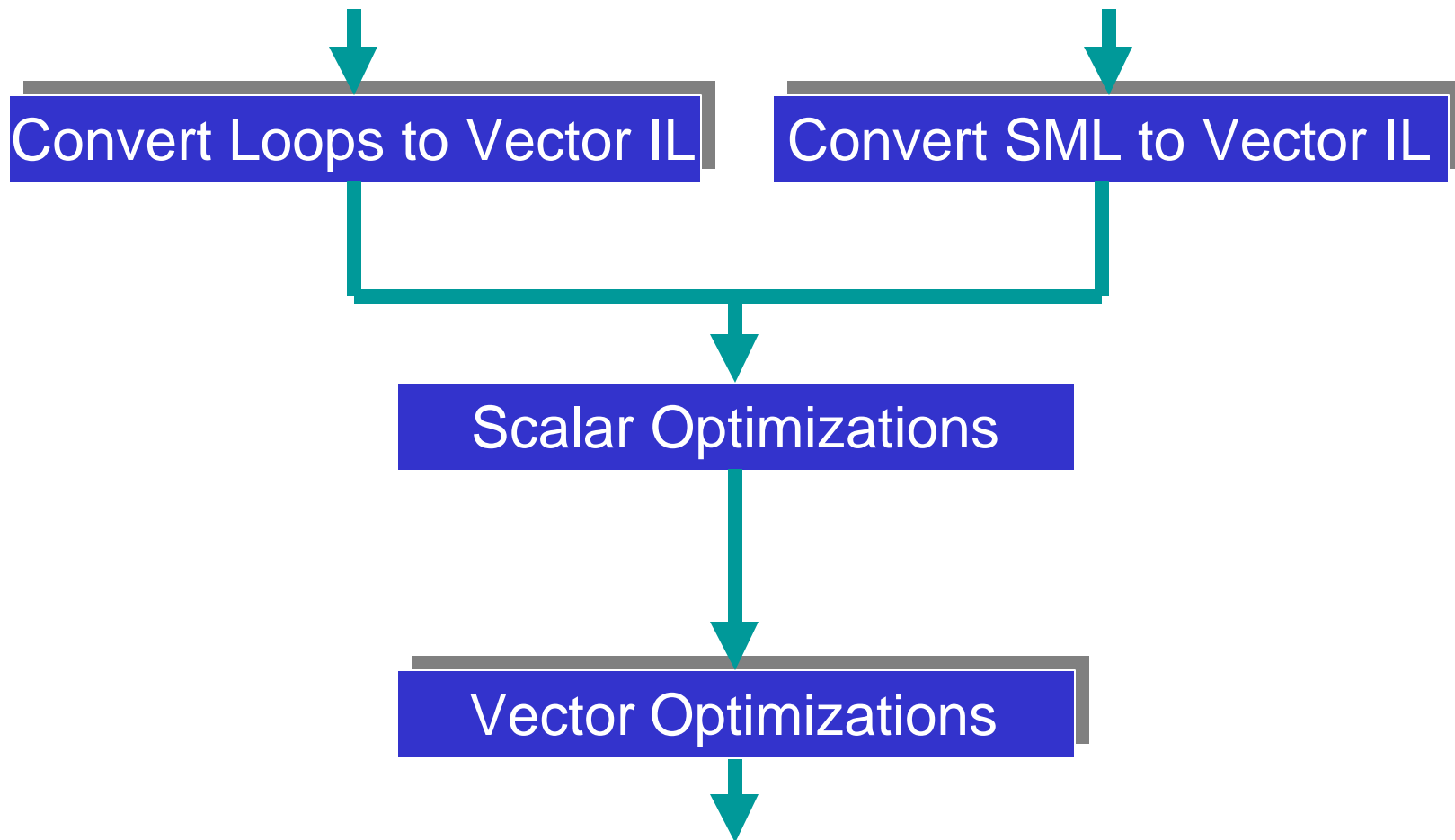
Core Plus Development

- ❑ **Randy Judd's implementation used as basis**
- ❑ **SKY's vectorizing compiler optimized most functions**
- ❑ **Some functions converted to SML (FFT, FIR)**
- ❑ **Simple "cloning" allowed additional data types**
- ❑ **Results in nearly 1,500 VSIPL functions**

Conventional Compiler



Vectorizing Optimization



Vector Optimizations

- Strip Mining**
- Function Joining**
- Function Chaining**
- Loop Inversion**

Performance

- ❑ **“Object Oriented” costs performance**
- ❑ **VSIPL .15 to .25 μ seconds/call slower than SML**
 - Not important for long vectors, or complicated functions like FFT, transcendentals
- ❑ **Takes time to retrieve address, stride and length from objects**
- ❑ **VSIPL**
 - `vsipl_vadd_f(a_view, b_view, r_view)`
- ❑ **SML**
 - `vadd(a, 1, b, 1, r, 1, length)`
- ❑ **Additional overhead in maintaining objects**

Lessons Learned

- ❑ **Automatic Vectorizer useful for quick performance gain**
- ❑ **Focus effort on code sections with the biggest gain**
- ❑ **Resulting performance equivalent to writing in assembly**