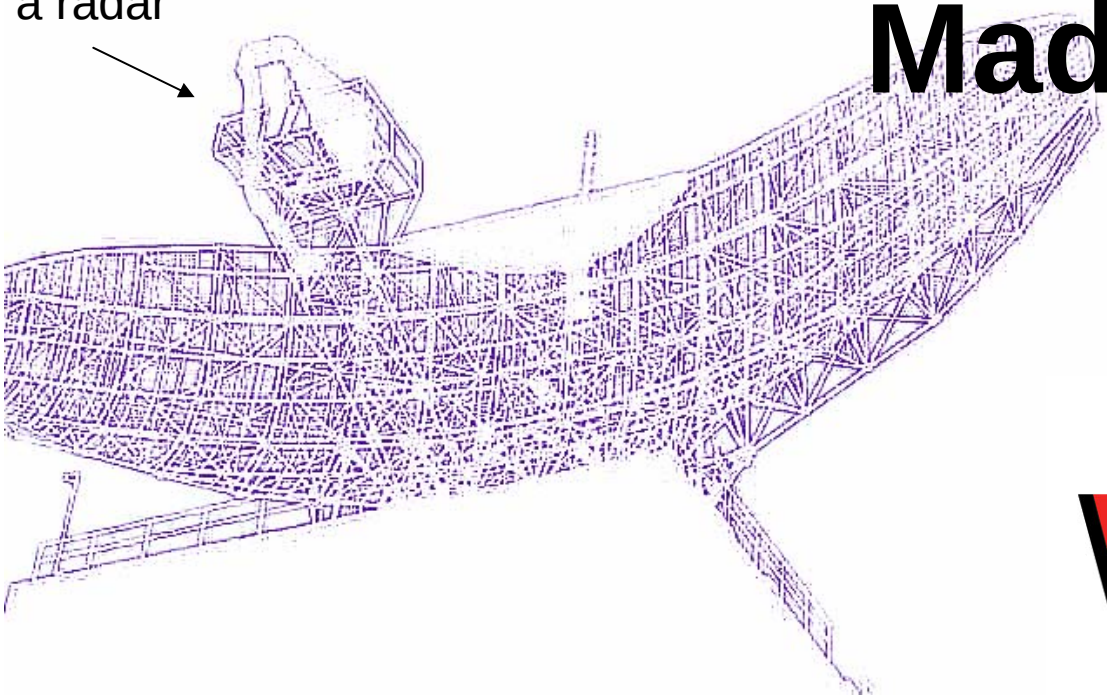
Verari systems software™

Pulse Compression Made Easy with VSIPL++



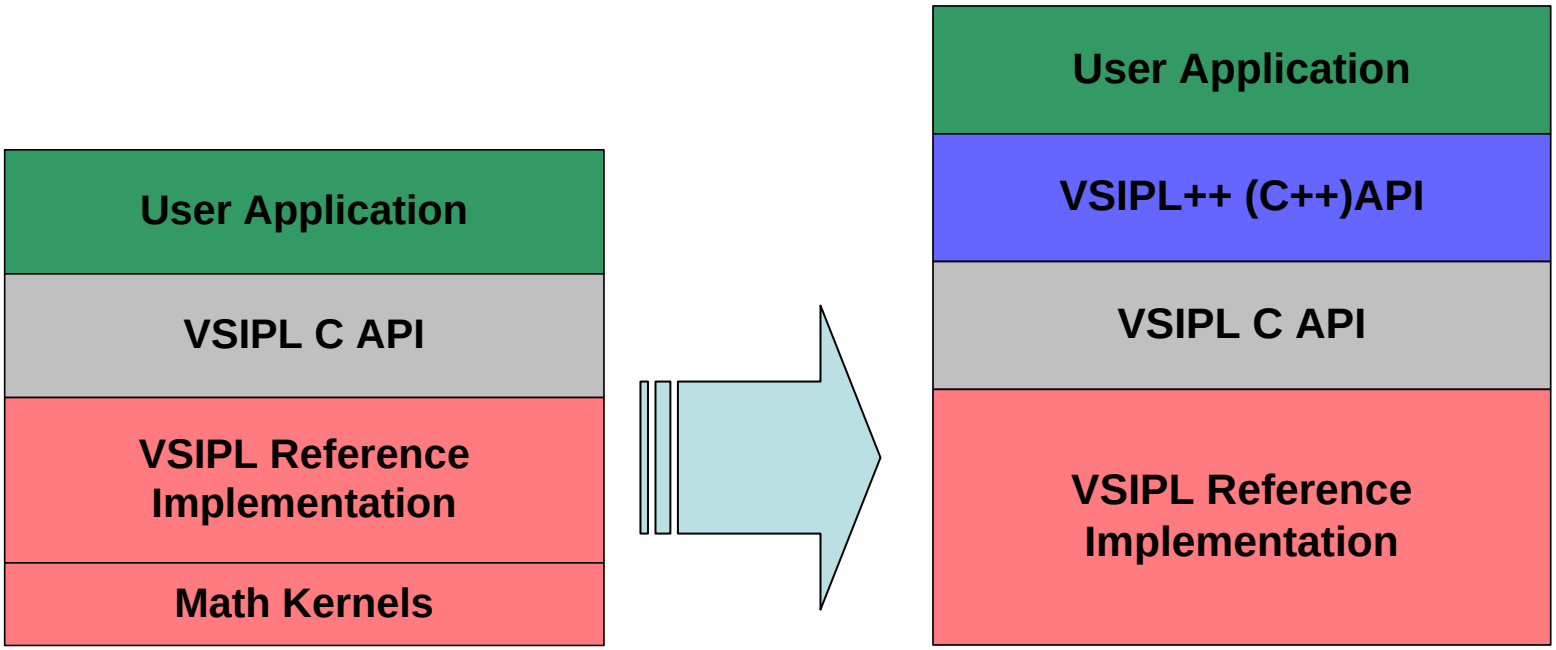
Verari systems software™

Funded Under SBIR
Topic OSD03-022 (OSD/AF)
“High Performance Object Oriented Software for Parallel Embedded Systems”

Formerly MPI Software Technology, Inc.

Verari systems software

VSIPL and VSIPL++ Reference Implementations

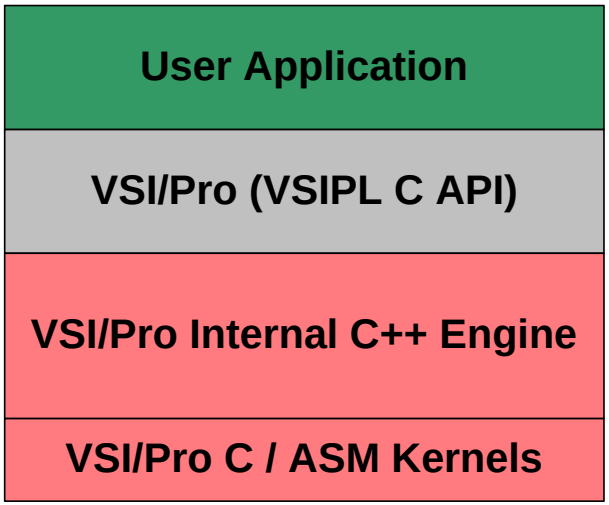


The VSIPL Reference Implementation

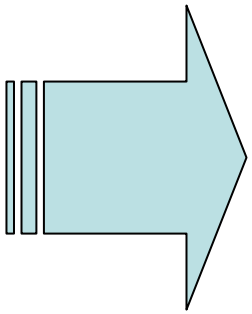
The VSIPL++ Reference Implementation
Builds upon the VSIPL Reference Implementation

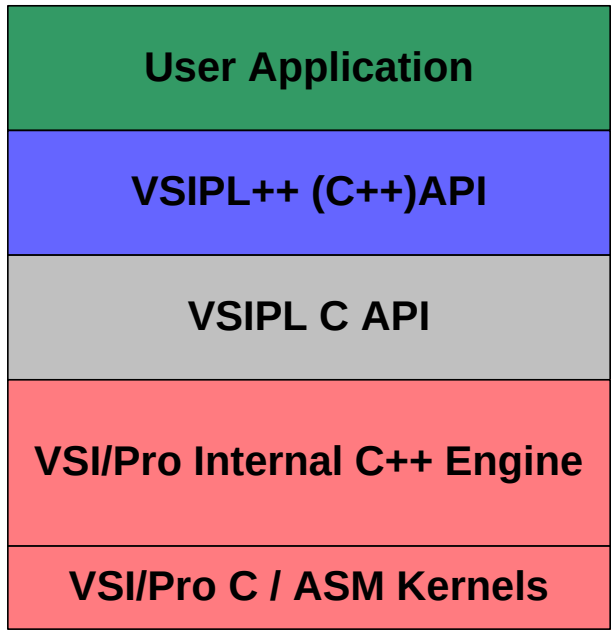
Verari systems software™

VSI/Pro Product and the VSI/Pro++ Prototype



Structure of VSI/Pro

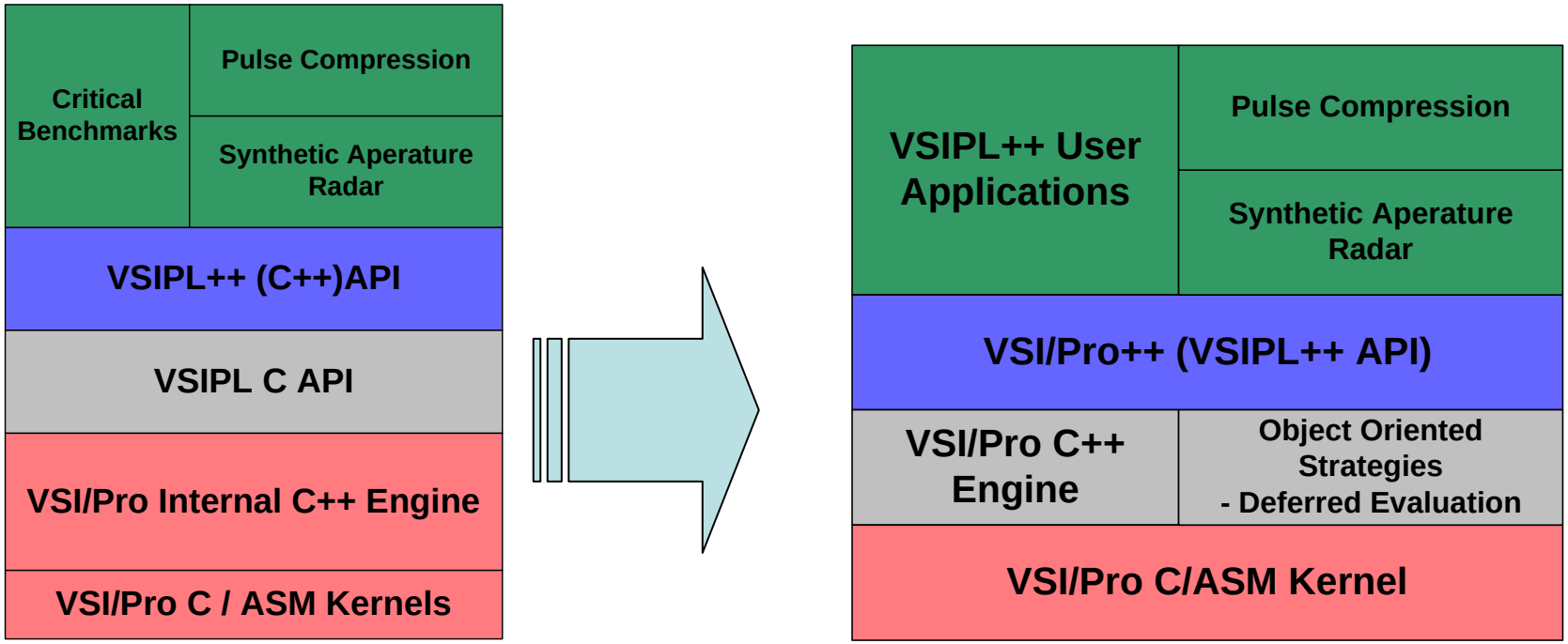




The VSI/Pro++ Prototype
Builds upon the VSI/Pro Product

Verari systems software™

Layered Approach versus a Pure Implementation



- What are the benefits of a Pure VSI/Pro++ Product.
- Having both API bindings available is a hidden benefit to programs that want to migrate their systems from VSIPL to VSIPL++ in phases.

Verari systems software™

Performance Comparison for 1024 Point Complex FFT

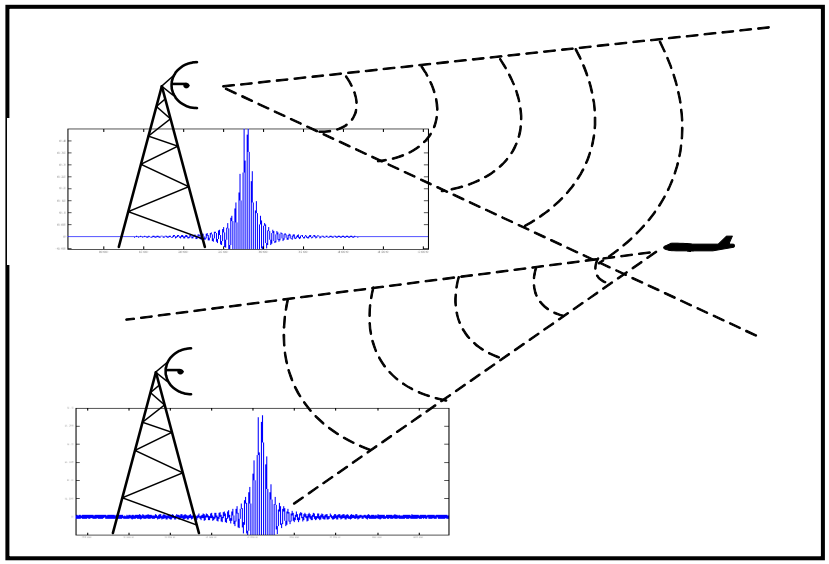
	Data Size	CCFFT by value	CCFFT by reference
VSIPL (VSI/Pro)	1024	does not apply	11.52 us
VSIPL++ (VSI/Pro)	1024	18.74 us	12.24 us
		multiple CCFFT by value	multiple CCFFT by reference
VSIPL (VSI/Pro)	1024 sets of 1024	does not apply	80 ms
VSIPL++ (VSI/Pro)	1024 sets of 1024	127.540 ms	82.350 ms

- Did not experience any significant overhead from layering the VSIPL++ API on top of the VSI/Pro API (See rightmost column).

Verari systems software™

Case Study: Pulse Compression

Pulse Compression works by distributing the energy in the outgoing Radar pulse over a larger span of time with one of a select number of waveform pulses that are generally known as chirp waveforms. This kind of filtering not only improves the accuracy of the measurements, but also rejects most kinds of ambient noise. The net effect is an improvement in resolution and decreased demand for peak power requirement in the signal generation equipment. A typical pulse consists of a short burst of frequency like the one shown here.



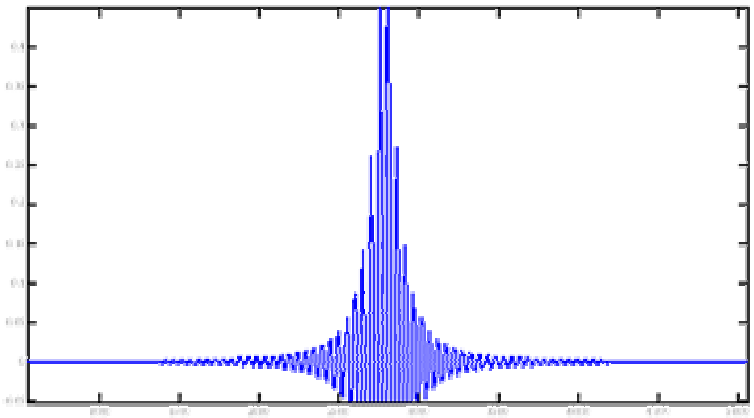
The digital signal processing functions that are associated with pulse compression applications typically use a complex FFT, a complex reference multiply, followed by an inverse complex FFT. Pulse compression, and FFT processing in general comprise a major portion of the processing load in state of the art radar systems.

Pulse Compression: The VSIPL way

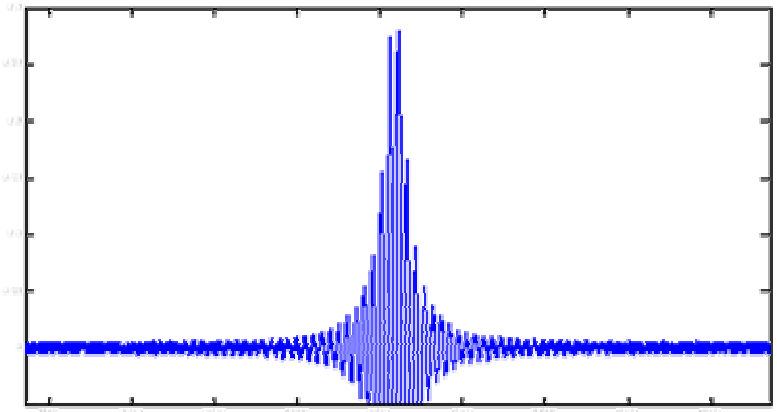
```
The pseudocode:
Create Vectors
Create Forward FFT object
Create Inverse FFT object

Create 3 temporary vector to hold intermediate frequency domain results.

Convert reference signal vector to the frequency domain:
Forward FFT( Ref Signal Vec, Temp Vec1 )

Convert Input signal to the frequency domain:
Forward FFT( Input Signal Vec, Temp Vec2 )

Multiply vectors in the frequency domain:
Vector Multiply( Temp Vec1, Temp Vec2, Temp Vec3 )

Obtain the inverse FFT:
Inverse FFT( Temp Vec3, Answer Vec )
```

Pulse Compression: The VSIPL++ way

```
The pseudocode:
Create Vectors
Create Forward FFT object
Create Inverse FFT object

Answer Vec = INV_FFT( FFT(Input Vec)*FFT(Reference Signal Vec));
```



Reference Signal



Noisy Return

Verari systems software™

Integrating Expression Manipulation into VSIPL++

Expression object strategies address the important problem of temporary copy proliferation that occurs as a result of operator overloading in C++.

Existing technologies that were studied -

- **PETE (Portable Expression Template Engine)**
  Developed at the Advanced Computing Laboratory at the Los Alamos National Laboratory
- **BLITZ++**
  The goal of Blitz++ is to provide a similar level of performance on par with Fortran 77/90
- **FACT! (Functional Additions to C++ through Templates and Classes)**
  A library that provides expression manipulation plus other functional programming language features not normally accessible in C++.

Verari systems software™

Observations from using VSIPL++

Benefits:

- **Concise code**
- **Readable**
- **Natural looking expressions**

Hazards:

- **Complex looking data types, may be helped in practice by typedefs**
- **General C++ concerns (e.g., possible to abuse the language)**