Rapid Portable Signal Processing Software Development Architecture

Kevin C. Tirko
Pennsylvania State University
Applied Research Laboratory

Abstract

The Pennsylvania State University's Applied Research Laboratory is continuing to develop an architecture independent signal processing software development capability. This capability permits rapid transition of algorithm design specifications and/or Matlab algorithm implementations into embedded target architectures. The C++ language and object-oriented techniques were chosen to promote software class development permitting a high degree of software reuse in future applications. The VSIPL middleware standard is used as the portable performance bridge to multiple target architectures.

The initial application to use this capability was a torpedo-embedded SONAR algorithm suite that performed data conditioning, signal correlation, signal detection, and target feature measurement. The algorithm design was transitioned during 2001 from an extensive Matlab testbed onto an embedded PowerPC architecture under VxWorks in approximately 4 months and was subsequently fielded in-water with success. Additional SONAR applications are utilizing the capability for field experiments in 2002 with productivity gains observed in excess of 400%.

The capability emphasizes operating system independence. All existing SONAR applications can execute under Unix, Linux, and VxWorks under the control of the appropriate Makefile or compilation switch. This allows application development to take place on a "convenient" system, perhaps awaiting Moore's Law solutions for the eventual embedded target.

The presentation will detail the primary motivations and features in the design of this capability. Specifically the many advantages of the C++ classes will be emphasized. The classes promote rapid code development, reduce debugging effort, and provide performance profiling and tuning of finished software. The classes also provide the interface to the VSIPL middleware standard library and hide all the details of VSIPL from the application software. As a result, application software has no responsibilities for initialization of VSIPL objects or VSIPL memory management which can be time consuming and problematic.