Integrating VSIPL Support in the Dataflow Interchange Format

Chia-Jui Hsu and Shuvra S. Bhattacharyya
Department of Electrical and Computer Engineering, and Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742, USA

Abstract

We have developed the dataflow interchange format (DIF) [2] and the associated DIF package for specifying and working with dataflow models for DSP systems. Our recent progress in the DIF project includes the *DIF-based porting approach* [2] for porting DSP designs across dataflow-based tools and the *DIF-to-C software synthesis framework* [3] for automatically generating C implementations from DSP system designs that are programmed in DIF. In this extended abstract, we present a new approach of using VSIPL [4] as an intermediate actor library for porting across different DSP design tools. Applying VSIPL in this manner builds on the increasing popularity of VSIPL as a standard DSP library, and eliminates the need to have actor mapping specifications between every pair of tools that we wish to port across. We also present an important new capability in DIF: DIF-to-VSIPL software synthesis. This capability augments the support of the DIF software synthesis framework and extends the reach of DIF-based interchange to the wide variety of platforms that support VSIPL.

The DIF Language and the DIF Package

The dataflow interchange format is a language for specifying and working with mixed-grain dataflow models for DSP systems. It provides a unique set of semantic features for specifying graph topologies, hierarchies, and dataflow-related as well as actor-specific information. The DIF package is the associated Java-based software package. It provides object-oriented dataflow representations, algorithm implementations, and infrastructure for porting and software synthesis. Figure 1 illustrates the methodology of using DIF to interface various dataflow models, DSP system designs, DSP libraries, dataflow-based DSP design tools, and their supported embedded processing platforms. The shaded areas in this figure show the new developments that involve the integration of VSIPL support into DIF.

VSIPL Integration in the DIF-based Porting

The idea behind the DIF-based porting approach is that except for actor information, a DIF specification for a DSP application represents the same semantic information regardless of which, if any, design tool is used to generate it, and furthermore, porting DSP applications can be achieved by properly mapping the tool-dependent actors, while transferring the dataflow semantics unaltered. The left shaded area in Figure 1 presents the porting mechanism that consists of *exporting* (exporting a design from a tool to DIF), *actor mapping* (converting attributes of the original actors to attributes associated with corresponding target actors), and *importing* (importing from DIF to another tool). We have developed the actor interchange format (AIF) [2] for specifying how to map actors across pairs of tools, and we have demonstrated the automation and efficiency provided by our DIF-based porting infrastructure.

One limitation of our original porting approach arises however when working with a large number of tools: when many tools are involved in the porting space, we need to specify the mapping information for each pair of tools. This requires effort and additional code that grows quadratically with the number of tools that are involved.

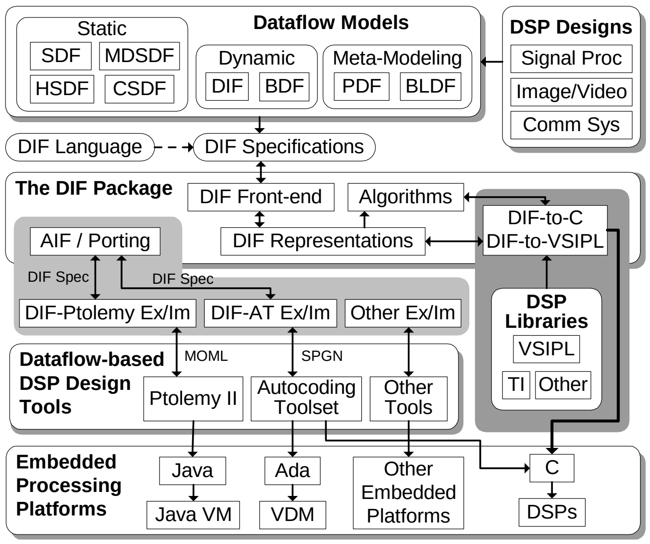

Figure 1: Methodology of using DIF.

VSIPL [4] is an open source, C-based API that provides various commonly used functions in vector and matrix computation, and many areas of signal processing. Motivated by the increasing popularity of VSIPL as a standard DSP library, we propose an enhanced DIF-based porting approach where VSIPL is integrated as an intermediate actor library, and the actor mapping mechanism operates by mapping “to” and “from” VSIPL.

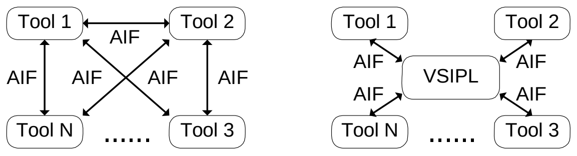

Figure 2: Original porting approach and VSIPL integration.

With this new configuration, as illustrated in the right part of Figure 2, we reduce the requirement of AIF specifications from $N(N\text{-}1)/2$ to N, and we also greatly facilitate the built-in AIF specifications for supported tools. Note that DIF supports both the original and the enhanced porting mechanisms. In particular, porting in a manner that bypasses VSIPL may still make sense when a small number of tools is involved or when direct mapping specifications are available or particularly easy to develop.

When employing VSIPL for porting purposes, its function prototypes are used as intermediate actor attributes. Issues arise due to certain special properties of VSIPL that have not been addressed in the developments of [2]. First, data types become explicit because VSIPL encodes data types (real/complex, dimension, and precision) into its naming conventions, and VSIPL functions are data-type-specific. This property does not generally hold for some tools, such as Ptolemy II [1], where actors support multiple data types. Therefore, we update the actor-block grammars in DIF and AIF and design special actor mapping strategies to include data type semantics. Second, VSIPL views encapsulate data dimension and size inside, whereas other tools, such as the Autocoding Toolset [6], may require specifying these parameters when using actors. As a result, we enhance the AIF semantics and actor mapping mechanism in order to access production and consumption rates for this purpose. Third, for certain computationally-intensive functions, VSIPL extracts parameters into separate objects, e.g., FFT and FIR objects. In DIF, such parameters are still specified as actor attributes associated with the VSIPL functions.

DIF-to-VSIPL Software Synthesis

We have developed the DIF-to-C software synthesis framework [3] to automate software implementation such that designers only need to program the dataflow semantics of DSP applications in DIF and associate actors with desired C library functions. This capability provides a vendor-neutral mechanism for linking coarse-grain dataflow optimizations with fine-grain hand-optimized DSP libraries and with DSP C compiler technology.

The DIF-to-C synthesis processes have been designed based on synchronous dataflow (SDF) semantics since SDF is the most mature model for dataflow-based DSP design. However, we observe that stream-based SDF semantics and array-based functions are sometimes inefficient when modeling multi-dimensional signal processing systems. Multi-dimensional synchronous dataflow (MDSDF) [5], where dataflow constraints are determined by *m*-dimensional production and consumption rates, has been developed previously to better accommodate multi-dimensional representation within the dataflow framework. One of the problems in developing MDSDF-based software synthesis is that efficient mechanisms are required to rearrange data between MDSDF semantics and one-dimensional memory layouts.

VSIPL adds a layer of abstraction involving the concepts of *blocks* and *views* to support portability across diverse memory and processor architectures. VSIPL blocks represent contiguous memory spaces where data is stored. VSIPL functions operate on views in a way that sets or subsets of data can be virtually arranged as vectors (1-D), matrices (2-D), or tensors (3-D). This feature makes VSIPL a particularly good match for integration with SDF and MDSDF semantics in software synthesis from DIF.

We have recently implemented multi-dimensional dataflow representations and scheduling techniques in the DIF package. We have also developed DIF-to-VSIPL software synthesis capability that supports both SDF and MDSDF by extending the original framework. Figure 3 illustrates the design flow of DIF-to-C/VSIPL software synthesis.

Given the buffer space (1- or *m*-D) for a dataflow edge computed by scheduling and buffering techniques, the DIF-to-VSIPL translation process creates a VSIPL block with size equal to the product of all dimensions. It also creates two VSIPL views (vector, matrix, or tensor views based on dimensions) associated with the block for source and sink actors (VSIPL functions). The *length* attributes of the views are decided by the production and consumption rates (1- or *m*-D), the *stride* attributes are determined by the buffer space (1- or *m*-D), and the *offset* attributes are adjusted between VSIPL functions based on the looped schedule (1- or *m*-D) and the token transfer rates.

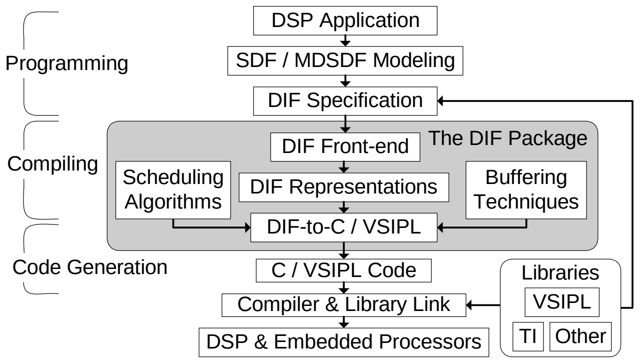

Figure 3: DIF-to-C/VSIPL software synthesis.

Acknowledgements

This research was supported in part by the U. S. Defense Advanced Research Projects Agency (DARPA) via the U. S. Army Aviation and Missile Command (Contract Number DAAH01-03-C-R236).

References

[1] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong, “Taming heterogeneity - the Ptolemy approach,” *Proceedings of the IEEE*, vol. 91, no. 1, January 2003.

[2] C. Hsu and S. S. Bhattacharyya, “Porting DSP applications across design tools using the dataflow interchange format,” In *Proceedings of the International Workshop on Rapid System Prototyping*, Montreal, Canada, June 2005.

[3] C. Hsu, M. Ko, and S. S. Bhattacharyya, “Software synthesis from the dataflow interchange format,” In *Proceedings of the International Workshop on Software and Compilers for Embedded Systems*, Dallas, Texas, September 2005.

[4] R. Janka, R. Judd, J. Lebak, M. Richards, and D. Campbell, “VSIPL: An object-based open standard API for vector, signal, and image processing,” In *Proc. 2001 IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol.2, pp.949–952.

[5] P. K. Murthy and E. A. Lee, “Multidimensional synchronous dataflow,” *IEEE Transactions on Signal Processing*, vol. 50, no. 8, pp. 2064-2079, August 2002.

[6] C. B. Robbins, “Autocoding toolset software tools for automatic generation of parallel application software,” technical report, Management Communications and Control, Inc., 2002.