PORTABLE JAVA BASED TOOL ENVIRONMENT FOR AUTOMATIC MULTIDIMENSIONAL CODE GENERATION

Maritza Rodríguez Martínez, Domingo Rodríguez

Electrical and Computer Engineering Department
University of Puerto Rico, Mayaguez Campus
Mayaguez, PR 00681-9042

Abstract

In this paper we present a tool environment for automatic code generation, in classic C language, of signal processing algorithms using Kronecker products algebra formulations. The environment is written as Java applets to be accessed on the internet. In essence, we implement a tool environment based on Java, to aid in the analysis, design, modification and implementation of multidimensional fast Fourier transform (FFT) algorithms on hardware computational structures. In particular, the environment becomes useful in the FFT algorithm development process, identifying similarities and differences in the mathematical formulation of variants, in Kronecker products form, of FFT algorithms and the associated computational performance analysis.

1. Introduction

This work presents the development of a tool environment, based on Java, to aid in the analysis, design, modification and implementation of multidimensional FFTs for signal processing applications in science and engineering. It was developed for automatic C and C++ code generation of FFT algorithms expressed in the mathematical language of Kronecker products algebra, and it has been used for code generation in areas such as synthetic aperture radar (SAR) computational signal processing and digital communications [1].

Kronecker products algebra, a branch of multidimensional multilinear algebra, is used by scientists and engineers as a mathematical analysis and synthesis tool to aid in the process of mapping FFT and other computational Kronecker based algorithms to particular machine hardware structures [2]. However, we can say that, in general, implementing fast Kronecker products based algorithms on computational structures for near optimal results is not a trivial matter. Several reasons can be provided to support this claim.

One reason is that, for a specific machine or hardware computational structure (HCS), many variants of a given algorithm (which can be viewed as forming a set, this set called here computational mathematics framework (CMF)), must be tried in order to reach a near optimal implementation (see Figure 1).

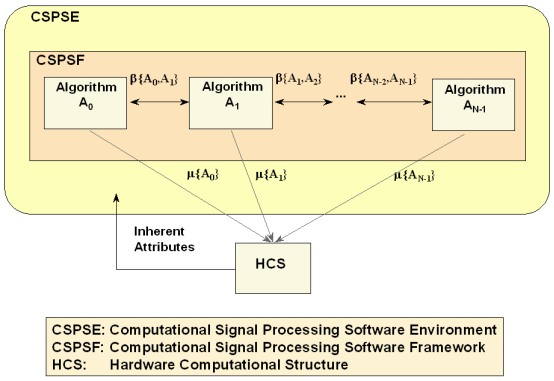


Figure 1. Mapping Variants to an HCS

Another reason is that, for a particular algorithm variant, obtained from a canonical formulation the computational performance of this algorithm changes from machine to machine, or from HCS to HCS, and it is desirable to identify on which machine the

algorithm variant performs best. This situation is described in Figure 2 below.

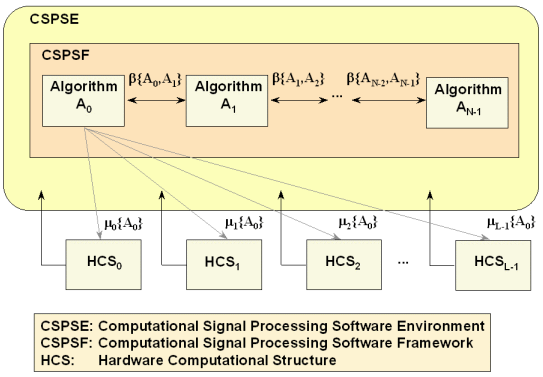


Figure 2. Mapping to Different HCS's

A third reason is that some observed inherent software and hardware attributes of a given machine or computational structure cannot always be expressed as parameters in mathematical expressions describing the Kronecker formulations of the FFT algorithms and, hence, near optimal computational performance cannot be claimed.

Regardless of the expressed shortcomings, Kronecker products algebra, with an associated set of permutation matrices called stride permutations, has been demonstrated to be very useful in the analysis and design of FFT algorithms, specially in endeavors such as structuring data flows, identifying inherent parallel and vector operations, obtaining FFT variants through the use of Kronecker products algebraic properties, and studying computational performance in specific computational hardware structures.

2. Kronecker Products Operators

2.1 Kronecker Products Algebra

The basic properties of Kronecker products algebra play a major role in the design and implementation of FFT algorithms. The formalism of Kronecker products notation can be used to keep track of the complex index calculations needed in FFT algorithms. The indexing property can be used to aid in program design and verification. Mathematical properties of Kronecker products can be used to guarantee the correctness of the implementation.

2.1.1 Kronecker Products

The Kronecker products between two matrices A and B of size $N \times N$ and $M \times M$ respectively is define as:

$$C = A \otimes B = [a_{N,N}]B \qquad (1)$$

$$C = A \otimes B = \begin{bmatrix} a_{0,0}B & a_{0,1}B & \cdots & a_{0,N-1}B \\ a_{1,0}B & a_{1,1}B & \cdots & a_{1,N-1}B \\ \vdots & \vdots & \ddots & \\ a_{N-1,0}B & a_{N-1,1}B & \cdots & a_{N-1,N-1}B \end{bmatrix}$$

which is a $MN \times MN$ matrix.

2.1.2 Kronecker Composition of the FFT

As demonstrated in [3], the FFT matrix, F_N, can be written as

$$F_N = (F_R \otimes I_S)T_{N,S}(I_R \otimes F_S)P_{N,R} \qquad (2)$$

where $N = R \cdot S$, I_N is a size $N \times N$ identity matrix. This formulation corresponds to the four-stage decimation-in-time Cooley-Tukey FFT.

2.1.3 Multidimensional FFT Theory

For arbitrary multidimensional Fourier Transform, let $N = N_1 \cdots N_t$, and let $X(a_1, \cdots, a_t)$ be a function of t variables. We can think of X as a function of one variable, and the t -dimensional Fourier Transform F_N becomes $F_{N_1} \otimes \cdots \otimes F_{N_t}$. The computation of $F_{n_1} \otimes \cdots \otimes F_{n_t}$ can be performed using a modification of the FFT that works independent of dimension. This generalized FFT allows an initial permutation of the input data and a more general class of twiddle factors. By modifying the initial permutation and the twiddle factors, the generalized FFT can be used to compute arbitrary multidimensional Fourier transforms[7].

3. Java Portability

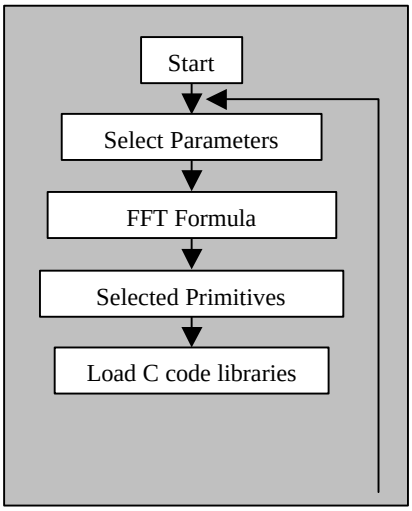
Java was designed to be an interpreted language. Because Java programs are interpreted rather than compiled, it is much easier to run them in a wide

variety of environments. Only the Java run-time system needs to be implemented for each platform. Once the run-time package exists for a given system, any Java program can run on it. Although the details of the Java run-time system will differ from platform to platform, but all interprete the same Java bytecode (highly optimized set of instructions designed to be executed by a virtual machine that Java run-time system emulates).Interpretation is the easiest way to create truly portable programs.

4. Java-FFT System Overview

The system flow diagram in Figure 3 shows the program development steps in the Java-FFT programming environment. The user can generate C code routines from the Kronecker based formulations by a selection of functional primitive operators in the application. The application translates the Kronecker expressions to a C code, which can then be viewed in a separate window. The application allows users to enter simple algorithm specifications, by editable text field, to generate complete Kronecker products formulations of one-dimensional and multidimensional FFT algorithms.

In the Java-FFT tool environment, to generate source code for a given Kronecker products formulation requires the following steps: Express the algorithm as a matrix-vector multiplication operation. This matrix is then factored into Kronecker products of sparse matrices, called functional primitives. Different Kronecker products compositions produce different mathematical formulations of one-dimensional and multidimensional FFT algorithms useful in obtaining implementation variants for targeted computational structures. The environment aids in identifying the most useful Kronecker products compositions and automatically generating C code.



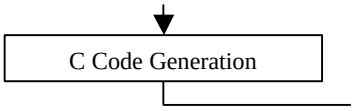


Figure 3. System Diagram.

5. Java-FFT Software Environment

The environment was developed using Java, a language for programming on the internet applet. Java applets are essentially applications that run inside a Java enabled browser, such as Netscape, Microsoft Explorer, or Hot Java. The purpose of using Java is the portability of the application. The user has remote access since he/she can run the application through an internet browser. We created a Java Applet with different buttons to select the desired composition for an FFT. At the present time the environment is comprised of the following components:

1. Primitive Operator buttons: the primitive operator buttons is a collection of different buttons, which allowed a user write a possible combination of Kronecker primitive operators such as:

$$F_N \otimes I_S, T_{N,S},\ I_N \otimes F_N,\ P_{N,S}$$

2. Text area: the text area shows the current combination selected by the user.

Clear button: the clear button helps to edit the selected combination and, when desired, it clears the text area.

3. Code button: when active, the code button shows a window containing the file with all C subroutines files.

In Figure 4 we show the Java-FFT client-server environment. The application has different files, with predefined blocks of C code that define the operators or Kronecker functional primitives. While the user selects (in the analysis mode) or composes (in the synthesis mode) the desired FFT computational Kronecker products formulation, the application translates the formulation to a C code, which contains the targeted subcoding.

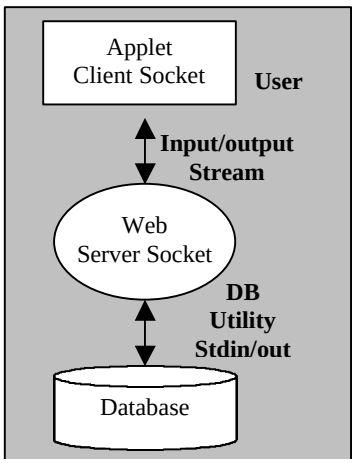


Figure 4. Client-Server

5.1 Java-FFT Tool Interface

Using an internet browser, a version of the applets interface appears as in Figure 5. The user is allowed to edit various parameters through the keyboard. The functional primitives are seen on the left side of the interface, to be selected in order to obtain the desired composition. The resulting composition is then written in the text area provided by the application.

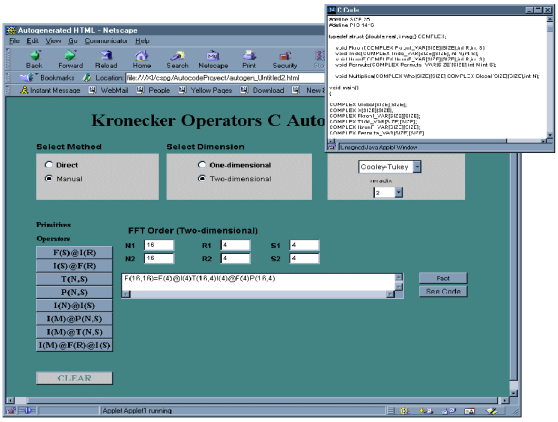


Figure 5. User Interface – Synthesis

Another version of the interface (the analysis mode) allows a user to produce a desired C-code of a selected FFT Kronecker products composition from a predefined set of FFT compositions conforming an FFT framework. The generated code will appear on a separate window as shown on Figure 6.

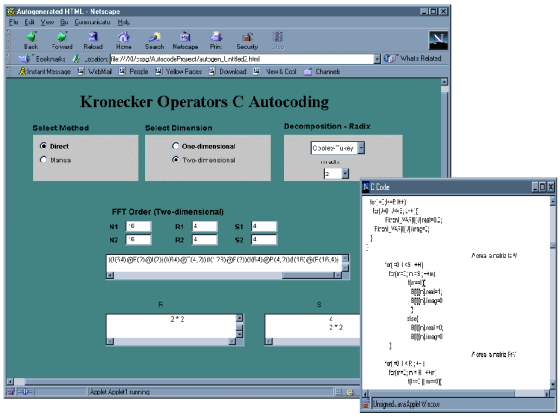


Figure 6. User Interface – Analysis

6. Conclusions

In this paper, we presented a Java programming tool environment for the automatic C and C++ code generation of multidimensional computational Kronecker products FFT mathematical formulations. This tool environment allows a user to analyze similarities and differences between FFT variants originated from a given FFT canonical formulation through the use of properties of Kronecker products algebra. The Java-FFT tool can be accessed remotely via the internet [4].

7. Acknowledgment

We acknowledge the contribution provided by the undergraduate student Ms. Joannie Madera in conducting the necessary software performance analysis. We thank Prof. Manuel Pérez-Quiñones for the suggestions provided to enhance this work.

8. References

[1] D. Rodríguez, “SAR Point Spread Signals and Earth Surface Property Characteristics (Invited Paper)”, SPIE's 44th Annual Meeting and Exhibition, Denver, Colorado, July 18-23, 1999.

[2] J. Johnson, R. Johnson, D. Rodríguez, R. Tolimieri, “A Methodology for Designing, Modifying, and Implementing Fourier Transform Algorithms on Various Architectures.” Journal of Circuits, Systems and Signal Processing, Birkhäuser, Vol. 9, No. 4, 1990.

[3] D. Rodríguez, "Tensor Product Algebra as a Tool for VLSI Implementation of the Discrete Fourier Transform," IEEE ICASSP '91, Toronto, Canada, May 1991.

[4] John Glossner, Jesse Thilo, and Stamatis Vassiliadis . “Java Signal Processing: FFT with bytecodes”, 1998. http://www.cs.ucsb.edu/conferences/java98/papers/jfft.pdf