Perfromance Evaluation Methodology for Automatic C-Code Generation of Signal Processing Algorithms

Joannie Madera-Villanueva
Advisor: Dr. Domingo Rodríguez

Computational Signal Processing Group
Electrical and Computer Engineering Department
University of Puerto Rico, Mayagüez Campus
Mayagüez, Puerto Rico 00681-5000
jmadera@ece.uprm.edu

Abstract

In this paper, we present a methodology of the performance evaluation of source C-code produced using two types of automatic code generation tools. The generation tools have been studied in detail and their product have been analyzed from the point of view of code length, time perfomance, code legibility and explicitness, computational stability, and code portability and reusability. An evaluation methodology is proposed and tested using fast Fourier transform (FFT) algorithms as automatic generated product codes. This is an ongoing work and some initial recommendations are provided as a result of our findings through our research work.

1. INTRODUCTION

Very often in the area of Computational Signal Processing we are faced with the task of developing a proof of concept from a theoy or an idea through a basic application prototype (see Figure 1). The software involved is usually of considerable size and of sound technical background. This requires is most cases to test the fundamental concepts of the theory in high-level computation and visualization programming environment such as MATLAB [1]. Once the fundamental theory is understood and the algorithms have been developed and tested in the MATLAB environment, it is desired to optimize this algorithms in parameters such as speed, memory use, length of source code, etc., for a target hardware computational structure or digital signal processing unit. This process is very time consuming when perfomed mannually and it is often desired to identify techniques which can help in its automation. As a solution to this problem, some researchers are resorting to automatic code generation. This approach, however, has not been quite succefull due to the many unknowns which surface in the procedure and rendering it sometimes useless. In our work we have selected the MATLAB software package, with its C-Code compilation capability to study the automatic code generation process and try to shed some light into this difficult endeavor. By way of comparison, we have also chosen a Java-based tool developed in our Computational Signal Processing Group (CSPG) for the automatic C-Code generation of FFT algorithms via the Internet. We have narrowed our research to study of FFT code generation in order to have more control over the various parameters that we want to identify and understand in this process. Our methodolgy for performance evalution in presented in the of block diagrams in Figures 2 and 3 below. In Figure 4 we show a diagram of

the MATLAB C-Code generation and evaluation process.

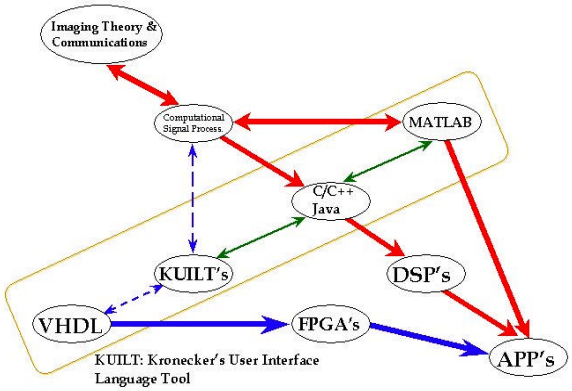

Figure No. 1: Proof of Concept Flowchart Through Application Prototype.

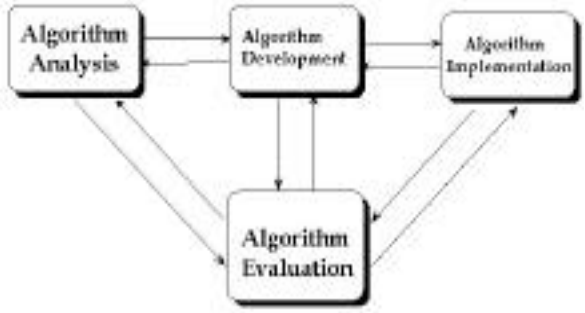

Figure No.2: Diagram of the Evaluation Methodology.

2. MATLAB AUTOCODING

The MATLAB Autocoding gives the opportunity to produce an executable mex-file from the m-file. The file extension is *.dll for Windows 95/98/NT and *.mex for Linux. During the process to produce this mex-file the Autocoding creates the libraries (*.h) and c-code (*.c or *.cpp). The user needs to remember to write the m-file in function form and use the version of 5.3 of MATLAB.

Figure No. 4 shows the process to obtain the mex-files and generated C-Code file wich will be used for the application prototype.

Basically, to reach at this point a person can start by writing an m-file in a notepad or editor/debugger. Use MATLAB to run the m-file and test the function created. Repeat this process until the fundamental concepts in theory coincide with the comptational results. Proceed, then to generate the mex-file using the MATLAB Autocoding and compilation environment. This is invoked through the prompt “mcc -x my_file”, in the MATLAB command window.

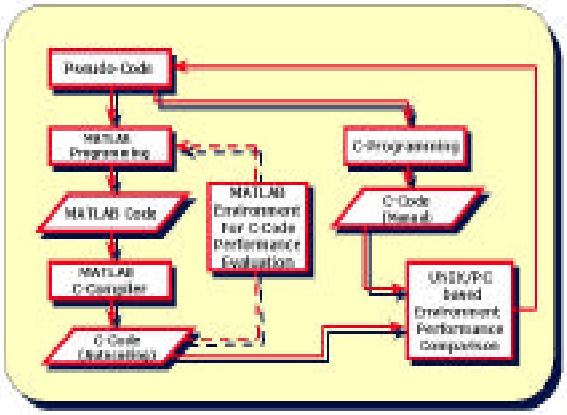

Figure No. 3 : Autocode Performance Analysis

The C-Code produced by MATLAB is tested using two avenues, the MS Visual C/C++ and Linux Compiler for C/C++. The code is tested for error(s) and warning(s) and an executable is produced. The mex-file produced is also tested in MATLAB. We now turn to the Java-base autocoding procedure for performance comparison on FFT algorithm C-Code generation.

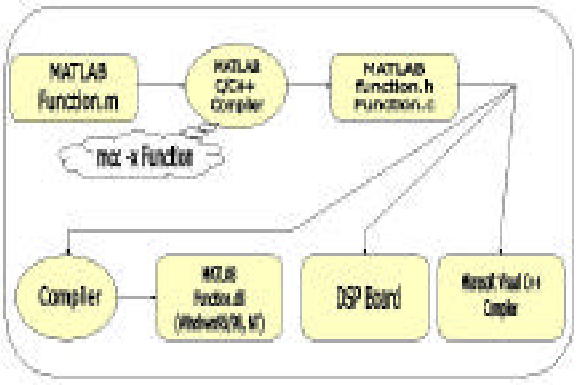

Figure No. 4: The MATLAB C-Code Generation and Evaluation Process

3. JAVA-BASED AUTOCODING

As we explained above the MATLAB software package has autocoding capabilities for producing C-Code. Figure 5 shows a an example of a computational environment developed by CSPG for image processing where it produces C-Code through the MATLAB utility.

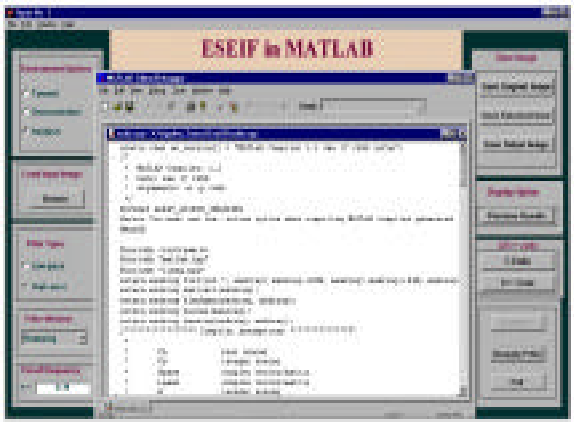

Figure No. 5 : Example of code created by MATLAB

In order to have more control over the parameters involved in the process of automatic C-Code generation, a Java-based tool was developed by CSPG [2] for FFT algorithms. The tool has two basic modes, an analysis mode and a synthesis mode. In the first mode (see Figure 6) the user is permitted to choose among a list of known mathametical formulations of FFT algorithsm such as Cooley-Tukey, Pease, Stockham, Korn-Lambiotte, etc. In the synthesis mode (see Figure 7), the user is allowed to composed a given FFT algorithm from basic factors called functional primitives.

4. CONCLUSION

As expressed at the outset, this interesting work is still on going and not final results have been obtained.

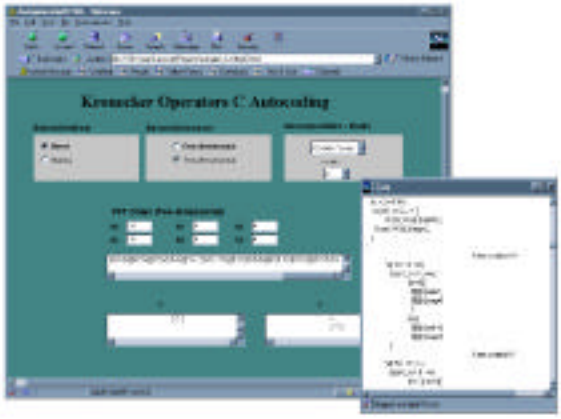

Figure No.6 : FFT-Java Enviroment (Analysis Mode)

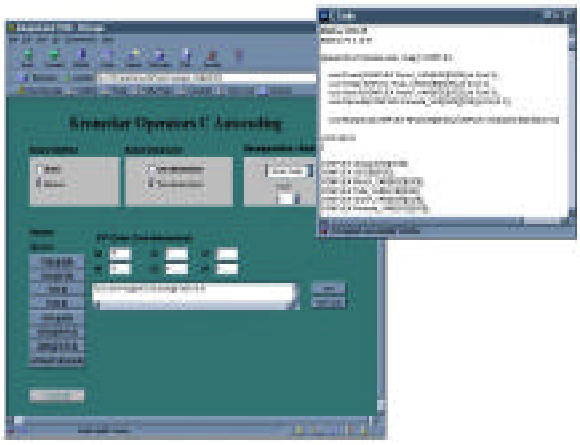

Figure No.7 : FFT-Java Enviroment (Synthesis Mode)

REFERENCES

[1] The MathWorks Inc., "MATLAB® the Language of Technical Computing", *http://www.mathworks.com*.

[2] M. Rodríguez, D. Rodríguez, "Java-Based Tool for Automatic Multidimensional FFT Code Generation," *The International Conference on Signal Processing Application & Technology*, (Orlando, Fl, USA), 1999.

APPENDIX: PARTIAL C-CODE GENERATED BY JAVA-BASED TOOL

```
/*****************************************/
/*main.c                                 */
/*Maritza Rodríguez Martínez             */
/*Computational Signal Processing Group  */
/* march-98                              */
/*CSPG-Dr. Domingo Rodríguez - Coordinator */
/*                                       */
/*Description: This function is the main function*/
/*of the generated code                  */
/*****************************************/
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define SIZE 1024
#define PI 3.1416

typedef struct {double real, imag;} COMPLEX;
   COMPLEX **getComplexMatrix(int axis);
   COMPLEX ** FkronI(int R,int S);
   COMPLEX ** Tride(int N,int S);
   COMPLEX ** IkronF(int R,int S);
   COMPLEX ** Permuta(int N,int S);

   void Multiplica(COMPLEX
              **Who,COMPLEX **Global,int N);

void main(void)
{

   COMPLEX **Global;
   COMPLEX **X;
   COMPLEX **FkronI_VAR;
   COMPLEX **Tride_VAR;
   COMPLEX **IkronF_VAR;
   COMPLEX **Permuta_VAR;

  int i,j;
  Global = getComplexMatrix(8);
  X = getComplexMatrix( 8);

       for(i = 0; i < 8 ; ++i)
        for(j = 0; j < 8 ; ++j)
           if(j==i){
            Global[i][j].real= 1.0;
             Global[i][j].imag= 0.0;
}
   else{
      Global[i][j].real =0.0;
     Global[i][j].imag= 0.0;
 }
 for(i = 0; i < 8 ; ++i)
 for(j = 0; j < 8 ; ++j){
     if(j==0){
      X[i][j].real=1.0;
       X[i][j].imag=0.0;}

    else{
     X[i][j].real=0.0;
       X[i][j].imag=0.0;
}
}

**********//**********//**********
**********//**********//**********
```