MEMSCOPT: A SOURCE-TO-SOURCE COMPILER FOR DYNAMIC CODE ANALYSIS AND LOOP TRANSFORMATIONS

Grigoris Dimitroulakos, Christakis Lezos, Konstantinos Masselos

University of Peloponnese, Department of Computer Science and Technology
Terma Karaiskaki, 22100 Tripoli, Greece
{*dhmhgre, lezos, kmas*}@*uop.gr*

ABSTRACT

In this paper, we present MEMSCOPT, a source-to-source compiler incorporated in a system level design tool chain for dynamic code analysis and loop transformations targeting memory performance optimization. MEMSCOPT is user interactive, supported by both Windows and Linux platforms and integrates with Visual Studio and NetBeans.

Index Terms— System level design, high-level synthesis, source-to-source transformations, dynamic code analysis

1. INTRODUCTION

Memory traffic has long time ago been proven to be a dominant factor for performance and power consumption of digital hardware systems [1]. Currently there is a large number of HLS tools delivering HDL specification from C-language application descriptions and recent works [2] have already started incorporating memory optimization before HLS. It has been proven that a better memory hierarchy can be synthesized when an optimized input specification is provided. The philosophy of our tools relies on interactively synthesizing a recipe of memory optimizing Source-to-Source (S2S) transformations for a specific application based on analysis results. Moreover, many programmers use Integrated Development Environments (IDEs) that provide comprehensive facilities to increase the programmers' productivity. With this in mind a S2S tool should also seamlessly integrate with popular IDEs.

In this paper, we present MEMSCOPT, an interactive tool assisting the optimization of memory hierarchy of a digital hardware system. MEMSCOPT was developed in the context of the FP7 ENOSYS project [3]. The aim of the ENOSYS project is to specify and develop a tool-supported design flow for designing and implementing embedded systems from UML input specification through seamless integration of high-level system specification, software code generation, hardware synthesis design space exploration and high level system description optimization. MEMSCOPT is part of the optimization environment that is developed to support the source code optimization phase in the ENOSYS project automated flow before conventional hardware synthesis and software compilation.

The tool offers two major facilities: 1) Pre-transformation dynamic code analysis and 2) Semi-automatic application of S2S transformations. Semi-automatic comes from the fact that the designer has to build up a script of transformations, based on analysis results, that is automatically realized by MEMSCOPT and XMSIM [4] tool. XMSIM is another tool developed in the same context that simulates the performance and power consumption of a given memory hierarchy for a given application's input specification in C language. There are MEMSCOPT's versions available for both Windows and Linux operating systems that integrate with MSVS and NetBeans IDEs correspondingly. Even though MEMSCOPT can execute from command line, this paper presents its usage with the user interface.

The paper is organized as follows: section 2 illustrates the user interface mainly from the interactive than the command line environment aspect while section 3 describes the demonstration example. Finally, section 4 summarizes the work and future actions.

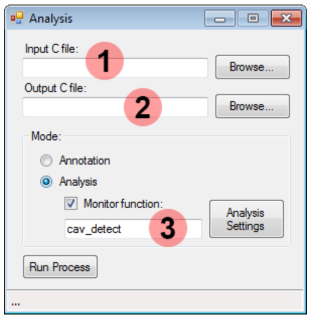
2. SYSTEM DESCRIPTION

MEMSCOPT's core is a command line application that acts on a single C file, each time it's called by the user. In order to extend MEMSCOPT's usability, the development of a friendlier graphical interface is decided. The graphical user interface (GUI) was developed using the C# programming language. The GUI is a separate application that can be seamlessly glued on top of MEMSCOPT and do away with the command line.

MEMSCOPT has 2 operational modes, one for each of its two major facilities: 1) Analysis mode and 2) Transformations mode. In the analysis mode, the tool embeds the analysis results directly into the application code using a Special Annotation Language (SAL). SAL is used for both documentation and analysis assistance. It is expressed inside C

*This work has been partially supported by the FP7 ENOSYS (FP7 ICT-2009.3.4 248821) project, funded by the European Community.

comments (Fig. 1b) and with it the user can interact with the tool by providing input regarding analysis. Currently, SAL includes directives for loop naming, loop iterator identification and loop count. Figure 1a shows the interface of the analysis mode: The user can browse and select 1) a C code input file, 2) where to save the analysis output file and 3) the function to monitor. If the user doesn't provide a function name the whole number of application loops are monitored or the user may provide the function name he wish to focus the analysis.



Mode: Analysis [✓]
Monitor function: [✓] cav_detect

(a) Analysis interface.

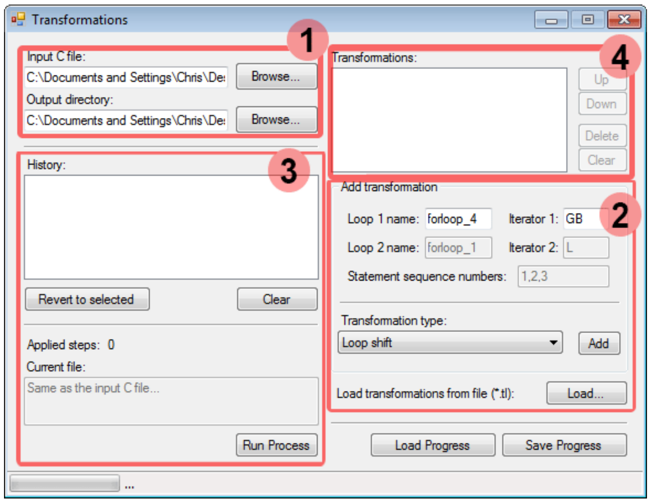
```
...

/* @forloop: forloop_0 */
/* @iterator: k */
/* @#iterations: 5 */
for ( k=-GB;k<=GB;++k ){
    tot+=Gauss[abs(k)];
}

...
```

(b) Special Annotation Language.

Fig. 1: Dynamic code analysis.



Input C file: C:\Documents and Settings\Chris\De:
Output directory: C:\Documents and Settings\Chris\De:
Loop 1 name: forloop_4 Iterator 1: GB
Loop 2 name: forloop_1 Iterator 2: L
Statement sequence numbers: 1,2,3
Transformation type: Loop shift
Applied steps: 0
Current file: Same as the input C file...

Fig. 2: S2S loop transformations.

Transformation of the code is an iterative process where each step applies one or multiple transformations. Figure 2 shows the interface for the transformation mode: It includes fields for: 1) Setting the input file and the output transformations' directory, 2) Selecting a transformation from a transformation pallete and configuring it, 3) Executing, saving and restoring the transformation steps made so far or backtracking to specific steps and 4) Manipulating the transformations of each step. Currently there are 7 types of transformations supported by MEMSCOPT: loop shift, loop extend, loop reversal, loop fusion, loop interchange, loop fission, loop normalization, loop reorder, loop switching, loopscopemoveforward and loopscopemovebackward. Finally, the whole number of transformation steps applied, result in a transformation script which is an XML file recording the initial input file and the full detail of each transformation step.

3. DEMONSTRATION

To demonstrate the functionalities of MEMSCOPT compiler we perform dynamic code analysis and apply a series of loop transformations on an image processing application called cavity detector [5]. Cavity detector consists of a series of loops, making it ideal for use as a demonstration of MEMSCOPT.

The series of loop transformations applied to cavity detector consist of a number of steps containing one or multiple transformations each. An output file is generated for every one of these steps so that analysis can be performed at any point of the process. Finally, the whole progress is saved in an XML file for future use.

The primary objective of this demo is to display the capabilities of the tool. Optimization of the code depends on the transformations provided by the user.

4. CONCLUSION AND FUTURE WORK

In this paper, we present a source-to-source compiler for dynamic code analysis and loop transformations targeting memory performance optimization. Ongoing work considers: 1) support for more S2S loop transformations and 2) the improvement of the analysis with a more sophisticated loop weight function.

5. REFERENCES

[1] F. Catthoor, K. Danckaert, K.K. Kulkarni, E. Brockmeyer, P.G. Kjeldsberg, T. Achteren, and T Omnes, *Data Access and Storage Management for Embedded Programmable Processors*, Springer, 2002.

[2] J. Cong, P. Zhang, and Y. Zou, "Optimizing memory hierarchy allocation with loop transformations for high-level synthesis," in *Proceedings of the 49th Annual Design Automation Conference*, New York, NY, USA, 2012, DAC '12, pp. 1233–1238, ACM.

[3] FP7 ENOSYS project, *http://www.enosys-project.eu/*.

[4] T. Lioris, G. Dimitroulakos, and K. Masselos, "Xmsim: Extensible memory simulator for early memory hierarchy evaluation," in *VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on*, july 2010, pp. 375 –380.

[5] M. Bister, Y. Taeymans, and J. Cornelis, "Automated segmentation of cardiac mr images," in *Computers in Cardiology 1989, Proceedings.*, sep 1989, pp. 215 –218.