Master Thesis Proposal

Interprocedural Constant Loop Bound Propagation for Patmos Architecture

180.778 Seminar for Master Students in Computer Engineering

Jérôme Hue - 12123713

March 9, 2023

Contents

1 Problem Statement

Real-Time systems can be defined as those systems in which the correctness of the sytem depends not only on the logical result of the computation, but also on the time at which the results are produced [1]. Therefore, there are limitations on how code can be written for such systems.

The Patmos processor has been specifically designed and optimized for real-time systems, and support a code generation technique called single-path code [2]. This technique ensures that all program executions follow a single path of execution, regardless of runtime events [3]. To achieve this, loop bounds must be available so that the correct code can be generated to ensure that loops always iterate the same number of times (equal to the upper bound).

The Patmos compiler supports a pragma that can be used in C code to declare the lower and upper bounds of a loop, which can then be used to generate single-path code. However, when loop bounds are dependent on function arguments, loop bound pragmas cannot be used because they are not constant, which results in failed compilations.

2 Aim of the work & Expected results

A constant loop bound refers to a fixed number of iterations that a loop will execute during the program's runtime. This project seeks to address the issue mentioned above by enabling constant loop bounds dependant on a variable given as an argument to be propagated across function call boundaries.

The expected results of this project would be a modification to the Patmos compiler that allows loop bounds dependent on function arguments to be declared using the loop bound pragma. This would enable the generation of single-path code in such cases, and therefore allow the use of Patmos processors in a wider range of real-time systems.

3 Methodological approach

Literature review and familiarization with patmos compiler

Collect theoretical information related to this problem, in particular, explore the existing techniques for generating single-path code and the role of loop bounds in this process.

Become familiar with the patmos compiler, analyze its limitation in handling loops with bounds dependent on function arguments.

Design & Implementation

Develop a method for propagating constant loop bounds across function call boundaries. Implement the method in the Patmos' compiler.

Testing and Evaluation

Test the modified compiler on a set of sample programs / algorithms (e.g. TACLe benchmarks [4]). Compare the results with the unmodified compiler to demonstrate the effectiveness of the proposed method.

4 State-of-the-Art

In real-time computing the correctness of a system depends not only on the logical result of the computation, but also on the time which the result are produced. This means that a delay in producing the result can have severe consequences, often because of the physical impact of the sytem on its environment [1]. To address this, techniques such as Worst-Case Execution Time (WCET) analysis have been developed to ensure that the system can meet its timing requirements under all possible conditions. However, with the increase in program and hardware complexity, finding good WCET analysis became impractical. Indeed, research on WCET analysis focus on one of the following sub-problems : modeling the timing of operations being performed on modern processors or finding way to automatically identify and characterize the possible dynamic control-flow of a software [5].

To solve this issue, single-path code was introduced in [3] as technique to generate code that always execute on one single execution path, allowing for simple WCET analysis. I [6], the single-path transformation technique was described, allowing to transform any reducible control-flow graph into a graph with a single execution path. This led to the implementation and evaluation of this transformation in a compiler backend for the time-predictable processor Patmos in [7]. The evaluation on a real-world application shows that single-path code generation is a practicable strategy for the construction of time-predictable software components.

Further recent development includes the investigation of the performance impact of adding support for dual issue pipelines to single-path code [8]. They show how single-path code has the potential to make efficient use the additional pipeline because of its high instruction-level parallelism. In [4], the authors propose an instruction filter that ban be integrated into a processor that is not compliant with single-path code. The filter dynamically modifies the instruction stream to generate a single-path code effect by monitoring the processor state and control-flow, and issuing single-path instructions.

Overall, the single-path code has demonstrated its potential as a practical strategy for producing time-predictable software components in safety-critical hard real-time systems. This technique have often been criticized to be performance degrading due to its control-flow serializing nature [7]. However, it have shown improvements in the predictability of execution time, reducing the complexity of WCET analysis and making it easier to design and verify real-time systems.

5 Relevance to the Curricula of Computer Engineering

This thesis covers the topics of compilers and program analysis. Numerous courses are dealing with this topic in Computer Engineering, with the following graduate courses being the most relevant :

- 104.271 Discrete Mathematics VO
- 104.272 Discrete Mathematics UE
- 185.A04 Optimizing Compilers VU
- 182.713 Real-Time Systems VO

In particular, 104.271 and 104.272 provide theoretical background in terms of graph theory and algebra. 184.*A*04 focuses on theory and practice of program analysis and verification.

Bibliography

[1] J.A. Stankovic. "Misconceptions about real-time computing: a serious problem for next-generation systems." In: *Computer* 21.10 (Oct. 1988). Conference Name: Computer, pp. 10–19. ISSN: 1558-0814. DOI: 10.1109/2.7053.

[2] Martin Schoeberl et al. "Patmos: a time-predictable microprocessor." In: *Real-Time Systems* 54 (Apr. 1, 2018). DOI: 10.1007/s11241-018-9300-4.

[3] P. Puschner. "The single-path approach towards WCET-analysable software." In: *IEEE International Conference on Industrial Technology, 2003.* IEEE International Conference on Industrial Technology, 2003. Vol. 2. Dec. 2003, 699–704 Vol.2. DOI: 10.1109/ICIT.2003.1290740.

[4] Michael Platzer and Peter Puschner. "A Processor Extension for Time-Predictable Code Execution." In: *2021 IEEE 24th International Symposium on Real-Time Distributed Computing (ISORC).* 2021 IEEE 24th International Symposium on Real-Time Distributed Computing (ISORC). ISSN: 2375-5261. June 2021, pp. 34–42. DOI: 10.1109/ISORC52013.2021.00016.

[5] Peter Puschner and Alan Burns. "Guest Editorial: A Review of Worst-Case Execution-Time Analysis." In: *Real-Time Systems* 18.2 (May 1, 2000), pp. 115–128. ISSN: 1573-1383. DOI: 10.1023/A:1008119029962. URL: https://doi.org/10.1023/A:1008119029962 (visited on 03/09/2023).

[6] Daniel Prokesch, Benedikt Huber, and Peter Puschner. "Towards Automated Generation of Time-Predictable Code." In: *14th International Workshop on Worst-Case Execution Time Analysis.* Ed. by Heiko Falk. Vol. 39. OpenAccess Series in Informatics (OASIcs). ISSN: 2190-6807. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014, pp. 103–112. ISBN: 978-3-939897-69-9. DOI: 10.4230/OASIcs.WCET.2014.103. URL: http://drops.dagstuhl.de/opus/volltexte/2014/4609 (visited on 03/08/2023).

[7] Daniel Prokesch, Stefan Hepp, and Peter Puschner. "A Generator for Time-Predictable Code." In: *2015 IEEE 18th International Symposium on Real-Time Distributed Computing.* 2015 IEEE 18th International Symposium on Real-Time Distributed Computing. ISSN: 2375-5261. Apr. 2015, pp. 27–34. DOI: 10.1109/ISORC.2015.40.

[8] Emad Jacob Maroun, Martin Schoeberl, and Peter Puschner. "Compiling for time-predictability with dual-issue single-path code." In: *Journal of Systems Architecture* 118 (Sept. 1, 2021), p. 102230. ISSN: 1383-7621. DOI: 10.1016/j.sysarc.2021.102230. URL: https://www.sciencedirect.com/science/article/pii/S1383762121001594 (visited on 03/09/2023).