If-Conversion To Reduce Worst Case Execution Time

Soma Niloy Ghosh[1], Lava Bhargava[2],and Vineet Sahula[3]
[1,2,3]Department of Electronics and Communication Engineering ,
Malaviya National Institute of Technology, Jaipur, India.
{2016rec9053[1], $lavab$[2], $vsahula.ece$[3]}@mnit.ac.in

***Abstract*—Worst case execution time(WCET) is a necessary parameter for scheduling a task in a system.WCET analysis provides a priori information about the worst possible execution time of a code before scheduling it in the system.If-conversion is an optimization that converts control dependence into data dependence. It removes branches and increases instruction level parallelism and thus performance. In this article we propose to perform if-conversion targeting WCET reduction. WCET is a key parameter in embedded system design.**

***Keywords*—If-conversion, WCET, compiler optimization.**

I. Introduction

WCET is an essential parameter for designing real-time systems. WCET of an application is the upper bound of its execution time for all possible inputs.WCET can be analysed statically or dynamically.Lesser the WCET lesser will be the resource demand of the application[4].For achieving this one of the approach is to optimize the application, which can be done at compilation time by applying different code optimization strategies.

If-conversion is a compiler optimization which converts control dependencies into data dependencies using using either full predication or partial predication(CMOVs).On one hand full-predication requires extra number of source operand for all instruction, but has largest performance improvement. On the other hand partial predication requires very less change in the existing architecture as only few instructions are provided with conditional moves[1].If-conversion gives more freedom for the compiler to reorder the instructions which could optimize the code and minimize the execution time.

Branch instructions are major limitations in exploiting instruction level parallelism(ILP). Predication removes branches which enables the compiler to perform if-conversion by eliminating branches and expose ILP. A major draw back of predication is, it executes nullified instructions which can degrade the performance.There can be many combinations of which branches to convert. In this paper we propose to solve this problem based on input from WCET analyser.

II. Methodology

Figure 1 describes the prototype to be implemented. The LLVM if-conversion transformation is included in the -O2 optimization level and can be separately requested using command line options. The SSA form simplifies implementation and allows existing SSA transformations to be used after if-conversion[2].

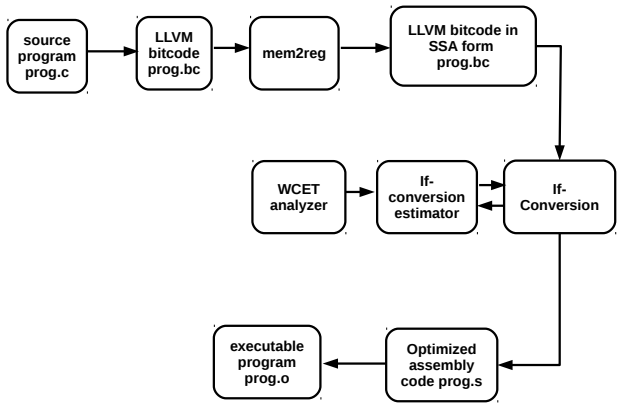

Fig. 1. Program compilation setup with WCET analyser

Not all branches are converted, only the branches that are in worst case execution path(WCEP) are converted and remaining are kept as it is.Therefore WCET is recalculated for every possible convertions. Key aspects of this technique are: (1) choose the branches present in the WCEP (2) eliminate the gotos and branches and insert logical variables to control the execution of the application and then calculate the WCET.

III. Experimental Results and analysis

Machine level if-conversion as done in LLVM -O2 phase is done in a straight forward manner. Predication is done easily as the instructions are already in target machine specific. If-conversion implemented in IR level can utilize the flexibility LLVM IR provides. Some optimizations that can be done in intermediate representations can be done after if-conversion but LLVM IR lacks predication[2]. As seen in figure 2 there is significant speed up after if-conversion in runtime of Mi-Bench[5] benchmark programs as compared to no compiler optimization.

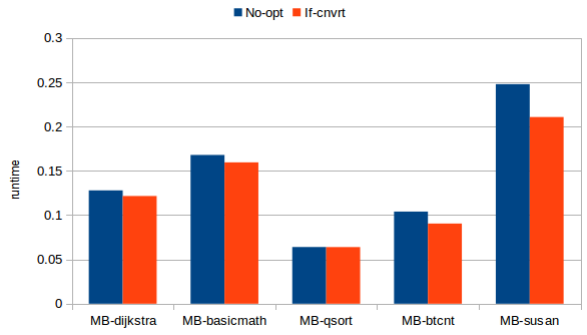

Fig. 2. Performance comparison of if-conversion with no optimization on different MiBench programs

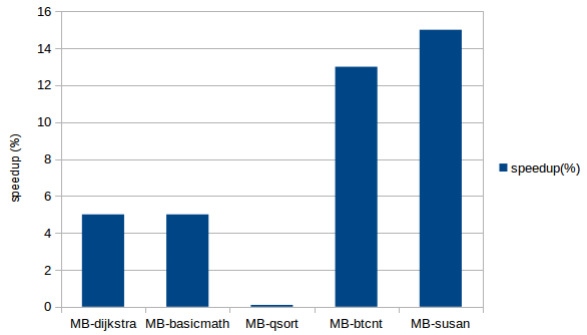

Fig. 3. Speed up in execution time of Mi-Bench programs after if-conversion

The speed up noticed is dependent on number of branches present in the application, the application having more number of branches will benefit the most due to if-conversion, more generically specifying, conversion of control dependence to data dependence. The *susan* program of MiBench benchmark suite is a large program with 15% speed up and has high potential for if-conversion if applied in IR level with other optimizations, *btcnt* program had speed up of around 13%, increase in speed of execution of code shows the minimization in worst case execution time of the applications. Similarly the speed up obtained by if-conversion on *dijkstra*, *basicmath* are 5% each and *qsort* application of same benchmark suite was 0.1%. As shown in figure 3.

Most of the optimizations is based on loop optimization, it could benefit if loop optimization and if-conversion are applied simultaneously to the basic blocks having both.

IV. Conclusion and Future work

Through our work we found promising gains due to the if-conversion if done in machine level.In future we will work on the conversions in IR level and apply other optimizations as well to achieve as tight WCET as possible. We also would study the effects of cache and code spilling due to the if-conversion optimization and its effects on WCET.

References

[1] S. A. Mahlke, R. E. Hank, J. E. McCormick, D. I. August, and W.-M. Hwu, *A comparison of full and partial predicated execution support for ilp processors*, in : Proceedings, 22nd Annual International Symposium on Computer Architecture (ISCA-22), 1995, pp. 138-149.

[2] A. Jordan, N. Kim and A. Krall, *IR-level versus machine-level if-conversion for predicated architectures*, in: Proceedings of the 10th Workshop on Optimizations for DSP and Embedded Systems (ODES13), 2013, pp 3-10.

[3] C. Lattner and V. Adve. *LLVM: A compilation framework for lifelong program analysis and transformation*. In International Symposium on Code Generation and Optimization (CGO), 2004, pp 75-88.

[4] H.Li, I.Puaut and E.Rohou *Traceability of Flow Information: Reconciling Compiler Optimizations and WCET Estimation*,in Proceedings of the 22nd International Conference on Real-Time Networks and Systems-(RTNS'14), 2014 ,pp 97-106.

[5] M.R.Guthaus,J.S.Ringenberg,D.Ernst,T.M.Austin,T.Mudge,and R.B.Brown, *Mibench: A free, commercially representative embedded benchmark suite*, in: Proceedings of the IEEE International Workshop on Workload Characterization, 2001.