Abstract

The design of many-core-on-a-chip has allowed renewed an intense interest in parallel computing. On implementation part, it has been seen that most of applications are not able to use enough parallelism in parallel register sharing architecture. The exploitation of potential performance of superscalar processors has shown that processor is fed with sufficient instruction bandwidth. The fetcher and the Instruction Stream Buffer (ISB) are the key elements to achieve this target. Beyond the basic blocks, the instruction stream is not supported by currents ISBs. The split line instruction problem depreciates this situation for x86 processors. With the implementation of Line Weighted Branch Target Buffer (LWBTB), the advance branch information and reassembling of cache lines can be predicted by the ISB. The ISB can fetch some more valid instructions in a cycle through reassembling of original line containing instructions for next basic block. If the cache line size is more than 64 bytes, then there exist good chances to have two basic blocks in the recognized instruction line.

The code generation for parallel register share architecture involves some issues that are not present in sequential code compilation and is inherently complex. To resolve such
issues, a consistency contract between the code and the machine can be defined and a
compiler is required to preserve the contract during the transformation of code. In this paper, we
present a correctness framework to ensure the protection of the contract and then we use code
optimization for verification under parallel code.

Reference

- Alex G., Avi M. Assaf S., Gregory S., Code Compilation for an Explicitly Parallel
  Register-Sharing Architecture, IEEE International Conference on Parallel Processing, 2007
- C. Lee, M. Potkonjak, W. H. Mangoine-Smith, “Mediabench: A Tool for Evaluating and
  Synthesizing Multimedia and Communications Systems”, 30th Annual ACM International
  Symposium on Microarchitecture, 1997
- D. Grunwald, H. Srinivasan, “Data Flow Equations for Explicitly Parallel Programs”,
  Fourth ACM SIGPLAN symposium on Principles and Practice of Parallel Programming, 1993
  Parallelism”, Proceeding of the 22th Annual International Symposium on Computer Architecture,
  1995
- David I. August Wen-mei W. Hwu Scott A. Mahlke, The Partial Reverse If-Conversion
- Dionisios N. Pnevmatikatos Manoj Franklin, Control Flow Prediction for Dynamic ILP
  Processors, Proceedings of the 26th Annual International Symposium on Micro-architecture,
  1993
- Guilin Chen, Mahmut Kandemir, Compiler-Directed Code Restructuring for Improving
  9, 2008
- J. Cong, Guoling Han, Zhiru Zhang, “Architecture and compilation for data bandwidth
  improvement in configurable embedded processors”, IEEE International Conference on
  Computer Aided Design, Proceedings of the 2005
- J. L. Henning, “SPEC CPU 2000: Measuring CPU Performance in the new Millennium”,
  Computer 33(7), 2000
  Special Issue on Billion-Transistor Processor, 30(9), 1997
- Marcos, Keali, “Exposing instruction level parallelism in the presence of loops”,
  Computation Systems Vol. 8 Number 1, pp. 074-085, 2004
- Noah Snavely, Saumya Debray, Gregory R. Andrews, Unpredication, Unscheduling,
  Unspeculation: Reverse Engineering Itanium Executable, IEEE Transactions on Software
  Engineering, Volume 31 Issue 2, 2005
- S. Muchnik, “Advanced compiler design and implementation”, Morgan Kaufmann
A Modern Parallel Register Sharing Architecture for Code Compilation

- Steve Car, Combining Optimization for Cache and Instruction-Level Parallelism, Proceedings of PACT’96, 1996
- Siwei Shen, David Flanagan, Siu-Chung Cheung, “An Extended Heuristic for Hyper-block Selection in If-Conversion”, Department of Electrical Engineering and Computer Science University of Michigan, Ann Arbor, Michigan 48109, USA

Index Terms

Computer Science

Parallel Computing

Key words

ILP

Multithreading

Fine-grained

Inthreads

ISB