Certified and Optimizing Bit Slicing Compiler

Mots clés :

- Directeur de thèse : gilles MULLER
- Co-encadrant(s) :
- Unité de recherche : Laboratoire d'informatique de Paris 6
- École doctorale : École Doctorale Informatique, Télécommunications, Électronique de Paris
- Domaine scientifique principal : Divers

Résumé du projet de recherche (Langue 1)

It is common knowledge that a modern computer manipulates 64-bit registers. Most programmers therefore have a deeply ingrained conception of the “atom of computation” being a 64-bit value, which could represent a number or a pointer for example. Software bit slicing [Pornin'01], also called “SIMD within a register” (SWAR) [Fischer'03], is a programming trick by which a 64-bit register is treated by the programmer as 64 1-bit registers. As a result, bitwise operations - for example, the logical negation of a 64-bit register - behave as a SIMD (“single instruction, multiple data”) instruction on 64 1-bit registers: we can exploit bit-level parallelism and therefore increase the throughput of some algorithms. This technique is particularly exploited in cryptography for its improved throughput on some cryptographic primitives [Biham'97, Canright'05, Azad'07] but also for its resistance against timing-attack [Kasper'09].

Résumé du projet de recherche (Langue 2)

Writing algorithms in a bit-sliced form is a tedious and error-prone task: in C or in assembly, programmers must implement their bit-level algorithms by manipulating 64 such bits at a time, thus obscuring their initial intent and losing the benefit of automated optimizations. Following an original proposition by X. Leroy, a first step is to design and implement a programming language for describing bit sliced algorithms. Due to the inherent bit-level nature of this formalism, we shall reuse concepts and techniques exploited by hardware description languages, such as the synchronous dataflow formalism [Biernacki'08]. The second step would be to generate optimized code, efficiency being usually measured in terms of “gate count” [Kwan'00]. To this end, one could both use standard techniques of Boolean evaluation [Knuth'05] while taking advantage of architecture-specific instructions, such as the Streaming SIMD Extensions (SSE) on Intel machines. The third step would be to specify the semantics of the description language and prove the correctness of the compiler in the Coq proof assistant. This work will build upon the “ssrbits” library [Blot'16] developed at LIP6. *(Objectives:)* - Implement a programming language suitable for describing bit-sliced algorithms - Implement & verify advanced optimization techniques for bit-slicing - Develop a machine-checked semantics of the language

Informations complémentaires (Langue 2)