Usuba,
optimizing and trustworthy bitslicing compiler

Darius Mercadier Pierre-Évariste Dagand
firstname.name@lip6.fr
Sorbonne Université, CNRS, Inria, LIP6,
F-75005 Paris, France

Bitslicing consists in reducing an algorithm to bitwise operations (`AND`, `OR`, `XOR`, `NOT`, *etc.*), at which point it can be ran with bit-level parallelism, viewing a n-bits register as n 1-bit registers, and a bitwise `AND` as n-parallel `AND` operators [1]. Bitslicing is thus able to increase performance by exploiting data-parallelism, while improving security by disabling cache-timing attacks – since a circuit runs in constant time. Bitsliced algorithms heavily benefit from SIMD extensions since their throughput is directly proportional to the size of the registers they use.

However, writing a program in bitsliced form is a tedious and error prone task, which produces a code that is hard to read, maintain, and optimize. To relieve the programmers from the burden of manually writing bitsliced code, we developed USUBA, a synchronous dataflow language producing bitsliced C code. The benefits of USUBA are threefold. First and by design, any software circuit specified in USUBA admits a bitsliced (and therefore efficient) implementation. Second, the informal description of symmetric cryptographical algorithms (by means of pseudo-circuits) directly translates into USUBA's formalism: as a result, one can effectively reason about – through a formal semantics – and then run the specification. Finally, USUBA generates optimized C code with SIMD intrinsics, without needing the programmer to write any architecture-specific code. Based on the AES implementation of Käsper and Schwabe [2], we designed a general model of bitslicing, called n-slicing, which gives the programmer fine-grained control over the structure of the C code generated by USUBA.

The codes generated by USUBA exhibits similar or slighly lower performances than hand-tuned C code on mainstream ciphers (like DES, AES, Serpent, Chacha20), while being able to be transparently ported on various vector extensiosn (SSE, AVX, AVX2, AVX-512) through a simple compilation flag.

References

[1] E. Biham. A fast new DES implementation in software. In *FSE*, 1997. doi:10.1007/BFb0052352.

[2] E. Käsper and P. Schwabe. Faster and timing-attack resistant AES-GCM. *CHES*, 2009. doi:10.1007/978-3-540-74735-2_9.