FACTORING A BINARY POLYNOMIAL OF DEGREE OVER ONE MILLION

Olaf Bonorden, Joachim von zur Gathen,
Jürgen Gerhard, Olaf Müller, and Michael Nöcker

September 18, 2000

Abstract. On 22 May 2000, the factorization of a pseudorandom polynomial of degree 1 048 543 over the binary field $\mathbb{Z}_2$ was completed on a 4-processor Linux PC, using roughly 100 CPU-hours. The basic approach is a combination of the factorization software BiPolAr and a parallel version of Cantor's multiplication algorithm. The PUB-library (Paderborn University BSP library) is used for the implementation of the parallel communication.

The approach. Polynomial factorization is a benchmark for the polynomial arithmetic components of computer algebra software. Many algorithms for polynomial multiplication, division with remainder and greatest common divisor have to be implemented and the crossover points between the various methods have to be determined, to finally combine them into a hybrid algorithm that works well for inputs of any size. Our parallelization is fine-grained, the Cantor multiplication routine running on 4 processors (see Wang 1996). Since many instances of a "small" problem have to be solved, the scalability of the parallelization depends heavily on fast communication.

The basic factorization algorithm. The sequential factorization algorithm used by our implementation is described in von zur Gathen & Gerhard (1996). The algorithm is an implementation of the general three-stage approach by Cantor & Zassenhaus (1981) with a blocking strategy, similar to the one in von zur Gathen & Shoup (1992); see von zur Gathen & Panario (2000) for an overview on polynomial factorization. Actually, we only parallelize the distinct-degree stage of the factorization. The cost of the initial squarefree factorization is negligible, as is the probability that substantial equal-degree factorization is necessary. The main task during the distinct-degree stage is the computation of *interval polynomials.* These are polyomials that have as factors all irreducible polynomials whose degree lies in a given interval and (almost) no factors of a degree larger than the upper bound of the interval. The computation of these polynomials requires many *modular squarings*. The preconditioned reduction modulo the polynomial remaining to be factored is based almost completely on costly multiplications, as is the asymptotically fast algorithm for computing greatest common divisors.

Parallel multiplication. Fast multiplication is the indispensable basis to treat polynomials of this size. Following its recursive structure, we parallelized the multiplication algorithm of Cantor (1989). The factorization is then done as follows. All processors execute the same sequential factorization algorithm synchronously, only joining forces when multiplying. Thus, every processor is completely aware of the context of the multiplication. For a multiplication no initial communication is necessary, only the result must be distributed to all processors. If the bandwidth is sufficiently large, this communication is dominated by the communication needed to collect the result on a single processor. Using this strategy, sequential state-of-the-art tricks such as Newton inversion and preconditioned multiplication and division are easy to parallelize.

We also implemented a second strategy based on the master-slave paradigm but it turned out to be inferior to the first strategy. The master processor executes the sequential factorization algorithm and distributes every multiplication to the other processors. These processors are unaware of the context of the computation and no CPU-time is wasted, but there is a substantial communication load. The implementation of this strategy was more complicated and thus prone to error, since many kinds of precomputed data have to be distributed. The fact that we do not need to distribute the inputs of the multiplication makes the first approach more efficient for all architectures that we tested.

The PUB library. The core of our implementation are three software packages: BiPolAr for sequential binary polynomial arithmetic, our parallel multiplication software, and the Paderborn University BSP library PUB. The latter library is an implementation of the theoretical BSP (Bulk Synchronous Parallel) model proposed by Valiant (1990). The efficiency of the implementation is largely due to the fact that the library offers several features extending the original BSP model. It is a flexible and easy-to-handle software tool which is available for several parallel computer architectures such as networks of workstations, cluster machines, or Crays. Detailed information, the source code, and the User Guide and Function Reference (Bonorden *et al.* 1999) are available from `http://www.upb.de/~pub/`.

The experiment. We ran our implementation on a Linux PC with 4 Pentium III processors, each rated at 500 MHz. The computer offers a total of 3 GBytes of memory. We factored a pseudorandomly chosen polynomial of degree 1 048 543 $= 2^{20} - 33$ with coefficients in $\mathbb{Z}_2$. We found 10 irreducible factors of degrees 23, 61, 239, 290, 4 555, 29 874, 59 193, 110 295, 123 712, 720 300. Thus interval polynomials had to be computed up to degree 360 150. In BiPoLAr an additional processor may be used to test the remaining polynomial for irreduciblity. The best known algorithm for this task substantially depends on arithmetical operations not covered by polynomial multiplication and thus has not been invoked in our test. Our implementation has an estimated speedup between 2 and 3.

Acknowledgements

This work has been partially funded by the Deutsche Forschungsgemeinschaft (DFG) under grant *SFB376 - Massive Parallelität, TP A1/A4*. We thank Ingo Rieping and the entire PUB team for their assistance.

References

Olaf Bonorden, Nicolas Hüppelshäuser, Ben Juurlink & Ingo Rieping (1999). *PUB-Library: User Guide and Function Reference*. Heinz Nixdorf Institute and Department of Computer Science, University of Paderborn, release 7.0 edition. URL `http://www.upb.de/~pub/`.

David G. Cantor (1989). On Arithmetical Algorithms over Finite Fields. *Journal of Combinatorial Theory, Series A* **50**, 285–300.

David G. Cantor & Hans Zassenhaus (1981). A New Algorithm for Factoring Polynomials Over Finite Fields. *Mathematics of Computation* **36**(154), 587–592.

Joachim von zur Gathen & Jürgen Gerhard (1996). Arithmetic and Factorization of Polynomials over $\mathbb{F}_2$. In *Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation ISSAC '96,* Zürich, Switzerland, Y. N. Lakshman, editor, 1–9. ACM Press. URL `http://www-math.uni-paderborn.de/~aggathen/Publications/polyfactTR.ps`. Technical report tr-rsfb-96-018, University of Paderborn, Germany, 1996, 43 pages.

Joachim von zur Gathen & Daniel Panario (2000). Factoring Polynomials Over Finite Fields: A Survey. *Journal of Symbolic Computation* . To appear.

JOACHIM VON ZUR GATHEN & VICTOR SHOUP (1992). Computing Frobenius maps and factoring polynomials. *computational complexity* **2**, 187–224.

L. G. VALIANT (1990). A bridging model for parallel computation. *Communications of the ACM* **33**, 103–111.

P. S. WANG (1996). Parallel Polynomial Operations on SMPs: an Overview. *Journal of Symbolic Computation* **21**, 397–410.

OLAF BONORDEN
Fachbereich 17 Mathematik-Informatik
Universität Paderborn
D-33095 Paderborn, Germany
`bono@upb.de`

JOACHIM VON ZUR GATHEN
Fachbereich 17 Mathematik-Informatik
Universität Paderborn
D-33095 Paderborn, Germany
`gathen@upb.de`

JÜRGEN GERHARD
Fachbereich 17 Mathematik-Informatik
Universität Paderborn
D-33095 Paderborn, Germany
`jngerhar@upb.de`

OLAF MÜLLER
Fachbereich 17 Mathematik-Informatik
Universität Paderborn
D-33095 Paderborn, Germany
`olafmue@upb.de`

MICHAEL NÖCKER
Fachbereich 17 Mathematik-Informatik
Universität Paderborn
D-33095 Paderborn, Germany
`noecker@upb.de`