Bethune, I and Gallot, Y 2015 Genefer: Programs for Finding Large Probable Generalized Fermat Primes. *Journal of Open Research Software,* 3: e10, DOI: http://dx.doi.org/10.5334/jors.ca

SOFTWARE METAPAPER

Genefer: Programs for Finding Large Probable Generalized Fermat Primes

Iain Bethune[1] and Yves Gallot[2]

[1] Project Manager, EPCC, The University of Edinburgh, GB
ibethune@epcc.ed.ac.uk

[2] Unaffiliated, FR
galloty@orange.fr

Genefer is a suite of programs for performing Probable Primality (PRP) tests of Generalised Fermat numbers $b^{2^n}+1$ (GFNs) using a Fermat test. Optimised implementations are available for modern CPUs using single instruction, multiple data (SIMD) instructions, as well as for GPUs using CUDA or OpenCL. Genefer has been extensively used by PrimeGrid – a volunteer computing project searching for large prime numbers of various kinds, including GFNs.

Genefer's architecture separates the high level logic such as checkpointing and user interface from the architecture-specific performance-critical parts of the implementation, which are suitable for re-use. Genefer is released under the MIT license. Source and binaries are available from www.assembla.com/spaces/genefer.

Keywords: Computational Mathematics; Number Theory; PRP Testing; Large Integer Multiplication; OpenCL; CUDA
Funding statement: Genefer has been developed with partial support from the Centre for Numerical Algorithms and Intelligent Software (NAIS), EPSRC grant EP/G036136/1.

(1) Overview

Introduction

Genefer is a suite of programs for performing Probable Primality (PRP) tests of Generalised Fermat numbers $b^{2^n}+1$ (GFNs). The code implements a Fermat primality test, computing a^{p-1} *(mod p)* for a candidate GFN p. It is a necessary, but not sufficient, condition for p prime that this test returns 1, so Genefer pseudo-primes must be subsequently tested for primality using a different program such as PFGW [1]. Genefer was originally developed by the second author, who managed a manual distributed computing effort between 2000 and 2004, which discovered several 100,000 + digit primes [2]. Since 2009, the PrimeGrid [3] volunteer computing project has used Genefer to extend the search to well over a million digits [4] and is currently searching for a world-record sized GFN prime. The current version 3.2.7 of Genefer includes contributions from a number of developers in the PrimeGrid community, and has been open-source since 2011.

Implementation and architecture

Genefer began as several monolithic C codes, each adapted for different CPU architectures (32-bit x86, x87 extended precision, and x86–64 with SSE2), with differing accuracy and performance characteristics. The accuracy manifests itself as a limit in the maximum value of b that can be tested for a given n before round-off errors are encountered (see **Table 1**). The limits are generated directly by the program and are checked at runtime. In version 2.3.0 the code was re-architected by the first author to provide a consistent user interface across all variants, and to define an API to separate the architecture-dependent portions of the code. The front-end contains common functionality such as command line argument parsing, the high-level logic which drives the Fermat test algorithm, checkpoint/restart support, as well as testing and performance benchmark. The performance critical implementation of the large-integer multiplication kernels are encapsulated by a C++ abstract class, and each different optimised version is an implementation of this.

Subsequently, many new variants of the multiplication step have been added, optimised for modern processor Instruction Set Architectures including AVX and FMA3/4, as well as GPUs using either CUDA or OpenCL. Originally based on Discrete Weighted Transforms (DWT) [5], recent versions have been used the z-Transform [6] for improved performance and better accuracy. In addition, support for BOINC [7] has been added, which enables Genefer to be used by many thousands of users transparently via the BOINC client installed on their home computers. Since version 3.2.0, multiple transform implementations may be compiled into a single executable, which may either be

Genefer	n = 19		n = 22	
Implementation	b limit	ms per mul	b limit	ms per mul
x87[1]	30,770,000	30.8	16,490,000	288
Default[1]	945,000	14.5	505,000	133
SSE2[1]	945,000	6.17	505,000	58.3
SSE4[1]	945,000	5.49	505,000	51.9
AVX[1]	945,000	3.66	505,000	35.8
FMA3[1]	945,000	3.35	505,000	32.6
CUDA (NVIDIA Tesla C2050)[2]	855,000	1.34	485,000	8.50
OpenCL (NVIDIA Tesla C2050)[2]	915,000	0.89	505,000	7.79
CUDA (NVIDIA Tesla K20m)[3]	825,000	1.05	480,000	6.09
OpenCL (NVIDIA Tesla K20m)[3]	915,000	0.54	505,000	4.19
OpenCL (AMD FirePro V7800)[4]	895,000	1.25	505,000	12.9
OpenCL (AMD FirePro D700)[5]	870,000	0.67	500,000	5.81

Table 1: Accuracy and performance of Genefer transform implementations. CPU hardware details: [1]Intel Core i7-4750HQ, [2]Intel Xeon X5650, [3]Intel Xeon E5-2670, [4]Intel Xeon E5620, [5]Intel Xeon E5-1650v2.

auto-selected at runtime based on the detected hardware, or selected by the user. A summary of the performance and accuracy of each of the transforms in Genefer 3.2.7 is given in **Table 1**. Note that the CUDA implementation is a DWT based on NVIDIA CUDA Fast Fourier Transform library (cuFFT) while the OpenCL version is a bespoke z-Transform.

The Genefer source code available from SVN contains a README with instructions on building the various different versions. Makefiles are provided for Windows, Linux, and Mac platforms, along with Visual Studio 2012 and 2013 solution files for building the GPU versions on Windows.

Usage

Every version shares the same command line interface, and supports three main modes of operation: interactive, where the user navigates a textual menu system; quick test, which tests a single candidate GFN specified via the `-q` flag; and batch mode, where a file is read containing a list of candidates to be processed.

For example, using the quick test (`-q`) option executing a probable primality test on a single candidate is as simple as:

```
$ ./genefer_macintel64 -q 2485064^4096+1
```

Giving the output:

```
genefer 3.2.8-dev (Apple-x86/CPU/64-bit)
Supported transform implementations: fma3 avx-
intel sse4 sse2 default x87
Copyright 2001-2015, Yves Gallot
Copyright 2009, Mark Rodenkirch, David
Underbakke
Copyright 2010-2012, Shoichiro Yamada, Ken Brazier
Copyright 2011-2015, Iain Bethune, Michael
Goetz, Ronald Schneider
Genefer is free source code, under the MIT license.

Command line: ./genefer_macintel64 -q
2485064^4096+1

Priority change succeeded.

Testing 2485064^4096+1...
Using FMA3 transform
Starting initialization...
Initialization complete (0.001 seconds).
Estimated time remaining for 2485064^4096+1 is
0:00:02
2485064^4096+1 is a probable prime. (26196 digits)
(err = 0.1562) (time = 0:00:02) 13:23:51
```

If several candidates are to be tested, they can be listed in a file of b, N pairs, one per line (some lines have been removed from the code output, for clarity), which is passed to Genefer as a command line argument:

```
$ cat input
2485064 4096
2485066 4096

$ ./genefer_macintel64 input
genefer 3.2.8-dev (Apple-x86/CPU/64-bit)
…
Start test of file 'input' - 13:41:11

Testing 2485064^4096+1...
…
2485064^4096+1 is a probable prime. (26196 digits)
(err = 0.1562) (time = 0:00:02) 13:41:13
```

```
Testing 2485066^4096+1...
…
2485066^4096+1 is a probable composite.
(RES=e011e285900ffee6) (26196 digits) (err =
0.1562) (time = 0:00:02) 13:41:15
```

The interactive mode may be accessed by running the binary without any arguments.

A "Generalized Fermat Prime Search" discussion forum exists at PrimeGrid (http://www.primegrid.com/forum_forum.php?id=75) where both the developers and users of Genefer are active. It is an excellent resource for asking questions and is searchable to find solutions to previously-encountered issues.

Quality control

Genefer contains a suite of inbuilt tests, which cover both prime and non-prime candidates for all relevant values of N – 32 to 524288 for CPU builds and 8192 to 4194304 for GPU builds. These are system tests that carry out an entire calculation from start to finish, and verify against a known reference result. Typically, these tests are run by developers as regression tests during the development of new versions.

Development versions of Genefer are identified by a version string with the '-dev' suffix. Prior to release, a set of acceptance tests are carried out in collaboration with PrimeGrid, which includes manual tests against known reference results, and integration tests where the release candidate app is used in a production environment and validated against the previous released version. These tests are carried out on all 3 supported platforms (Windows, Linux, Mac), for all transform types, and include use of the checkpoint/restart functionality. Once the acceptance tests are completed, the '-dev' suffix is removed and the release version is tagged in SVN. The released binaries are then distributed to PrimeGrid users via BOINC.

(2) Availability

Operating system

Runs on Windows (98 or later), Linux (2.6 or greater) and Mac OS X (10.5 or greater)

Programming language

Genefer is written in C++, with Intel vector intrinsics and long double types. The CPU version of the code can be compiled with gcc 4.9 or clang 3.6. The CUDA and OpenCL code are compiled with MS Visual Studio 2012 or later on Windows.

Additional system requirements

Genefer requires an x86 CPU. For the CUDA version of the code an NVIDIA GPU with compute capability 1.3 or above is required. For the OpenCL version, an OpenCL 1.0 device supporting the `cl_khr_fp64` or `cl_amd_fp64` extension (double precision floating point) is required.

Dependencies

Currently, the BOINC libraries are required to build genefer, although the code can be run in a stand-alone mode. For CUDA, the code may be compiled with CUDA 3.2 or greater. The distributed binaries are linked against version 5.5 or 6.0. For OpenCL, version 1.0 or greater is required.

List of contributors

Iain Bethune (Versions 2.3, 3.1 architecture, Mac & Linux porting)

Ken Brazier, Shoichiro Yamada (CUDA implementation)

Yves Gallot (Main author)

Michael Goetz (BOINC integration and CUDA)

Mark Rodenkirch, David Underbakke (x86/SSE and x87 assembly code)

Ronald Schneider (Linux porting)

Software location

Code repository

Name: Assembla.com

Identifier: http://www.assembla.com/spaces/genefer

Licence: MIT license

Date published: 23-Apr-2015 (version 3.2.7)

Language

English.

(3) Reuse potential

The principal aspects of Genefer which may be of interest for re-use are the large-integer multiplication routines, which are encapsulated within a C++ abstract class defined in '`Implementation.h`'. The API is very simple, and so it is easy to develop new implementations for integration in Genefer, or re-use these in other applications. A reference implementation is provided in the '`basicgenefer`' directory. For example, the second author is currently developing a new program 'cyclo' for testing another 'form' of Cyclotomic primes (http://primes.utm.edu/bios/page.php?id=4310), based on the Genefer transform code.

Another aspect of the code which might be re-used is the checkpoint/restart code. This is encapsulated within two C functions `read_checkpoint()` and `write_checkpoint()`, located in the file `common/check.cpp`, and has a simple and implementation-independent interface.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgements

Genefer has been developed and deployed with support from the PrimeGrid project, in particular John Blazek and Rytis Slatkevicius. We are grateful for many members of the PrimeGrid user community for contributing to testing prior to release of the code.

References

1. **Fougeron, J, Nash, C** and **Rodenkirch, M** OpenPFGW. Available at: https://sourceforge.net/p/openpfgw/.
2. **Dubner, H** and **Gallot, Y** 2002 Distribution of generalized Fermat prime numbers. Math. Comp., 71: 825–832. DOI: http://dx.doi.org/10.1090/S0025-5718-01-01350-3

3. **Bethune, I** 2015 PrimeGrid: A Volunteer Computing Platform for Number Theory, International Conference on Computational Mathematics, Computational Geometry & Statistics (CMCGS) 2015. DOI: http://dx.doi.org/10.5176/2251-1911_CMCGS15.43
4. **Bethune, I** and **Goetz, M** 2014 Extending the generalized Fermat prime search beyond one million digits using GPUs, Proceedings of the 10th International Conference on Parallel Processing and Applied Mathematics, PPAM 2013, Lecture Notes in Computer Science, 8384: 106–113. DOI: http://dx.doi.org/10.1007/978-3-642-55224-3_11
5. **Crandall, R** and **Fagin, B** 1994 Discrete weighted transforms and large-integer arithmetic. *Math. Comp.*, 62:305–324. DOI: http://dx.doi.org/10.2307/2153411
6. **Bruun, G** 1978 z-transform DFT filters and FFT's. IEEE Transactions on Acoustics, Speech and Signal Processing, Feb 1978, 26(1): 56–63. DOI: http://dx.doi.org/10.1109/TASSP.1978.1163036
7. **Anderson, D** 2004 BOINC: a system for public-resource computing and storage, Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, GRID '04. IEEE Computer Society, Washington, pp. 4–10. DOI: http://dx.doi.org/10.1109/GRID.2004.14

How to cite this article: Bethune, I and Gallot, Y 2015 Genefer: Programs for Finding Large Probable Generalized Fermat Primes. *Journal of Open Research Software,* 3: e10, DOI: http://dx.doi.org/10.5334/jors.ca

Published: 19 November 2015