

# Factoring Large Numbers with a Programmable Hardware Implementation of Sieving

Hea Joung Kim (kimmer@icsl.ucla.edu) and William H. Mangione-Smith (billms@ucla.edu)  
The UCLA Department of Electrical Engineering

## 1 Introduction

This paper develops and evaluates an architecture for high-speed number factoring on a configurable computing system based on field programmable gate arrays (FPGA)<sup>1</sup>. Currently, the primary interest in factoring large integers is to test the integrity of a number of cryptosystems, particularly the RSA public key system developed by Rivest, Shamir, and Adleman [4, 5]. The RSA algorithm can be used to provide encrypted transmission, authenticate the source of a transmission, and form the core of a number of other advanced security capabilities. Introduced in 1977, RSA relies on that fact that large integers are hard to factor. This paper presents an architecture that speeds up the sieving process 50~200 times when compared to a 200 MHz UltraSparc through parallel computing modules, pipelining within each module, and a special-purpose parallel memory system.

## 2 Factoring Algorithms

Interest in number factoring stretches back over many centuries. Fermat introduced the basic idea that  $n=pq$  or  $n=((p+q)/2)^2 - ((p-q)/2)^2$  where  $n$  is being factored. Assigning  $a=(p+q)/2$  and  $s=(p-q)/2$ , the  $n=((p+q)/2)^2 - ((p-q)/2)^2$  can be rewritten as  $n=a^2 - s^2$ . From that equation,  $a$  can be chosen from  $n^{0.5} < a < n$ , and if  $a^2 - n = s^2$  is a perfect square,  $n$  can be factored using  $\gcd(a+s, n)$ . From the algorithms of Fermat to the number field sieve (NFS) [16], factoring algorithms share a similar core. Most factoring algorithms quickly search for numbers  $t$  that are highly composite of prime numbers raised to even powers; i.e. a sieve is applied to find these perfect square numbers. The algorithm used for this paper is the multiple polynomial quadratic sieve (MPQS) which was the fastest known algorithm until the number field sieve. MPQS appears to remain the most efficient for medium sized numbers (up to ~350 bits). The sieving methods are quite similar for both MPQS and NFS. MPQS tries to find  $a \equiv b^2 \pmod N$ . The  $a$ 's, which are found by sieving, are composed of multiple powers of prime numbers from a list of approximately 524,338 primes and one or two primes less than  $2^{30}$  in magnitude. This set of prime numbers has been identified in advance, and is used repeatedly for all applications of MPQS. Since there are many possible  $a$ 's that can be found, this process can be executed in parallel with different polynomials for selecting the initial roots. Thus, the multiple polynomials can be executed separately in hundreds of machines on the Internet [20]. In April 1994, the RSA-129 number was factored by Atkins, Graff, Lenstra, and Leyland with the help of many others unused computing resources. The algorithm used for factoring RSA-129 was MPQS.

## 3 Sieving

We have developed a parallel hardware sieve for a specific polynomial. We have focused on the process of sieving through a single array. Our hardware is several orders of magnitude faster than any known implementation running on a general-purpose computer.

The low sieve (primes < half array size) can be optimized the most with our hardware techniques. The sieving task is relatively simple. Initially, the sieve array is cleared so that all entries are zero. Next, the main sieve passes the low sieve the values for  $ibegin$ ,  $iend$ , sieve array size, and the logarithm of the prime. The  $ibegin$  value is used to select the initial roots and prime. Then using the prime and roots, the entire sieve array is traversed. The contents of the array are read and the logarithm of the prime is added and then written back. All the additions of logarithm of the primes are 8-bit numbers. When the top of the sieve array is reached, the process stops.

The reason for adding the logarithm of the prime is to find  $a$ . Clearly,  $a$  is a composite of the powers of primes and so using the logarithm of prime is sufficient for finding the set of  $a$ 's.

The medium sieve works like the low sieve except for the fact that the primes are greater than half the sieve array size and less than the size of the entire array. Thus, in the sieving process the array is accessed either once or twice. The big sieve also sieves the array but has at most one hit since the roots have to be negative to allow for the prime number that is greater than the size of the array to hit a location in the array.

## 4 Mojave Configurable Computing Platform

The Mojave configurable computing project has been ongoing at UCLA since 1995 [15, 22], with a primary focus of accelerating key image analysis applications through the use of hardware that is modified while an application executes. The number factoring work has targeted the third generation Mojave computers, which is a interface to an i960 PCI board. The platform contains five Xilinx XC4085XL-3.

The hardware platform provides four computing nodes – 4 FPGAs. The computing nodes are configured through the fifth FPGA, which also manages the interface bus and all external IO. A 96-signal programmable ring-bus connects each of the computing FPGAs. Each computing node has four banks of local SRAM that are 16-bits wide and 256k words deep. Thus, systems can be built which provide access to a single bank of SRAM that is 64-bits wide, two banks that are 32-bits wide, etc. The i960 board from Cyclone Microsystems contains 32 MB of DRAM, which is used to store the prime array and sieve results.

Component costs for this hardware are approximately \$3000 US dollars as of December 1998, though component costs for FPGAs have been decreasing rapidly.

### 4.1 Sieving logic

In the sieving process, the sieve array is addressed by increments of primes. The fact that the sieving is done by increments of primes is significant. For any prime number, the SRAMs can be independently addressed to read the data in, add the logarithm of the prime, and write back the results. There will be no conflict where the multiple addresses address the same SRAM due to the prime numbers. Figure 1 shows an example, where the prime being used is three.

#### 4 Memory Banks per FPGA

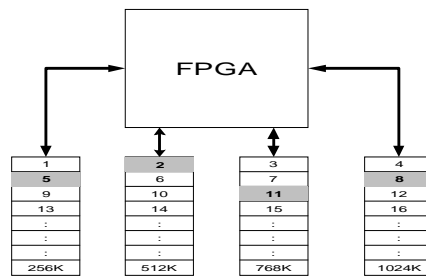


Figure 1. Prime interval addressing for root = 2 and prime = 3

With this technique, the low prime-sieving rate is increased by a factor that is proportional to the number of independently addressable SRAMs. In the case of the Mojave, the performance increase is by a factor of 4 per FPGA. If more I/Os are available on an FPGA, the speed up factor will increase with the number of SRAMs.

#### 4.2 Using Multiple Mojave FPGAs

Further improvements of the sieving process can be made using the other 4 FPGA on the system board. The low prime sieve can be sped up by factors that are near linear to that of the number of independently addressable SRAMs. However, the medium and big prime sieves are not improved by independently addressable memories. To improve the entire sieving process, spreading it over all sixteen memory modules on a single board can increase the size of the sieve array.

The sieving functions take the roots and primes and check if it indeed there is a location in the array the can be hit. If there is a hit the location is read, logarithm of prime is added, and written back to the SRAM.

### 5 Performance Results and Discussion

The hardware sieves were designed in Verilog using the Synopsys synthesis system and Xilinx backend CAD tools. The current system is accessed through a personal computer running Windows 95 over the PCI bus.

Simulations show that Sparc5 (microSparcII @110MHz), Sparc20 (SuperSparcII @75MHz), and Ultra2 (UltraSparc @200MHz) takes 110ms, 80ms, and 57.9ms, respectively to perform 85,000 sieve operations or sieveops. The performance is for a low sieve with iend-ibeg=55 using a sieve length of  $2^{20}=1,048,576$ .

A single FPGA on the Mojave board performs 85,200 sieveops in 1.92ms. The layout for the circuit used shows that neither the CLBs nor the external IO pins are close to complete utilization. Furthermore, the current layout is highly irregular. This reflects the fact that no optimization has been done on either the Verilog or the placement. As a consequence, we are confident that a significantly higher clock rate can be achieved. This avenue has not been pursued as the current design manages to saturate the SRAM interface; thus higher clock rates would only introduce wait states. We are in the process of moving to memory components that are significantly faster than the 30ns access parts that we are currently using. An FPGA can perform 4 sieve operations (4 addresses per clock) in 3 clock cycles (read, add, write). Therefore, at 33MHz,  $(85,200 * \frac{1}{4} \text{ parallel} * 3 \text{ cycles} * 30\text{ns}) = 1.92\text{ms}$  per sieveop.

We can further improve the performance of sieving in hardware because we have 4 FPGAs available for computing. Thus, there is another speed up factor of 4. As for the medium and big sieves, the gain comes from caching the prime and root arrays while spreading the work over the 4 FPGAs. Note that if the memory sizes were larger we would spend more time in the low prime sieve and thus achieve higher overall performance. Assuming the extreme case where the low sieve has an array size that is greater than largest prime number in the prime set, the speed up factor would be 16. The factor of 16 comes from the fact that all 4 SRAMs per FPGA are being utilized. In the worst case where the sieve array is smaller than the smallest prime, the speed up factor will be 4 due to the 4 processing elements being used.

#### 5.1 FPGA Critical Paths

The critical path is the time from generating a new address with a relatively large root and large prime. The new address is generated by addition, which has to be routed to one of the 4 independent addressing units. The breakdown is as follows: 6.1ns for addition, 17.77ns for routing and multiplexing, and 7.1ns for other logic and routes. We plan to further reduce this time to the point where the limiting critical time is not that of the FPGA logic or routing but the access time to the SRAM. Once the overall performance limited by the access to the SRAM there will be no reason to consider ASIC implementations. In sieving operations, the FPGA is essentially performing as well as any possible ASIC. If we are able to improve the critical path timing to <10ns, we would be able to perform the same 85,200 sieving operations in a third of the time (~0.66ms). For the low prime sieve, we have a factor of 4 further improvement by using a 5 FPGA board with 4 processing elements.

### 6 Conclusion and Future Work

The hardware implementation provides further speed up in the number factoring algorithm. The overall process can be sped up by a factor between 4 and 5. Amdahl's rule applies here because the sieving operation, which takes 80% of the time, can be minimized to zero but the 20% still remains. With this type of speed up, the time that it takes to break RSA-129 is brought down to 2 months. This calculation assumes normalization where the same numbers of identical CPU resources are substituted with FPGAs. Furthermore, this hardware sieving with Number Field Sieve will factor the RSA-129 in 2-3% of the time it took to factor RSA-129 using MPQS [23]. RSA130 was factored using 10% of compute time used to factor RSA129 using MPQS. So a hardware implementation will bring the fraction down to ~2%.

Further optimizations will be made to reduce the critical path of the logic and the routing. The reduction will force access time to the SRAM to become the limiting factor. This would result in a system that doesn't require an ASIC given that faster logic and critical path hardware will not be any faster due to the memory access time limitation. Thus, the FPGA performs as well as an ASIC.

## 7 Reference

- [1] C. Ebeling, D. C. Cronquist, and P. Franklin, "Rapid - Reconfigurable Pipelined Datapath," Proc. of Field Programmable Logic, 1996.
- [2] E. Mirsky and A. DeHon, "MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources," Proc. of Field-Programmable Custom Computing Machines, Napa, CA, 1996.
- [3] C. Pomerance, "The Quadratic Sieve Factoring Algorithm," Advances in Cryptology: Proceedings of EUROCRYPT 84, 1985.
- [4] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. New York: John Wiley & Sons, 1996.
- [5] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*: CRC Press, 1996.
- [6] D. Atkins, M. Graff, A. K. Lenstra, and P. C. Leyland, "The Magic Words are Squeamish Ossifrage (RSA public key cryptography)," Advances in Cryptology - ASIACRYPT'94, 1995.
- [7] G. Estrin, "Organization of Computer Systems: The Fixed-plus Variable Structure Computer," Proceedings of the Joint Western Computer Conference, 1960.
- [8] G. Estrin, B. Bussell, R. Turn, and J. Bibb, "Parallel Processing in a Restructurable Computer System," *IEEE Transactions on Electronic Computers*, pp. 747-755, 1963.
- [9] G. Estrin and R. Turn, "Automatic Assignment of Computations in a Variable Structure Computer System," *IEEE Transactions on Electronic Computers*, pp. 755-773, 1963.
- [10] W. H. Mangione-Smith and B. Hutchings, "Configurable Computing: The Road Ahead," Proc. of Reconfigurable Architectures Workshop, Geneva, 1997.
- [11] W. H. Mangione-Smith, B. Hutchings, D. Andrews, A. DeHon, C. Ebeling, R. Hartenstein, O. Mencer, J. Morris, K. Palem, V. K. Prasanna, and H. A. E. Spaanenburg, "Seeking Solutions in Configurable Computing," *IEEE Computer*, vol. 30, pp. 38-43, 1997.
- [12] W. H. Mangione-Smith, "Application Design for Configurable Computing," *IEEE Computer*, vol. 30, pp. 115-17, 1997.
- [13] W. Mangione-Smith, "Configurable Computing: Concepts and Issues," Thirtieth Hawaii International Conference on System Sciences, 1997.
- [14] M. J. Wirthlin and B. L. Hutchings, "Improving Functional Density Through Run-Time Constant Propagation," Proc. of Field Programmable Gate Arrays, Monterey, CA, 1997.
- [15] K.-G. Chia, H. J. Kim, S. Lansing, W. H. Mangione-Smith, and J. Villasenor, "High Performance Automatic Target Recognition Through Data-Specific VLSI," *IEEE Transactions on VLSI*, vol. 6, pp. pp. 364-372, 1998.
- [16] A. K. Lenstra, J. H. W. Lenstra, M. S. Manasse, and J. M. Pollard, "The Number Field Sieve," Proc. 22nd Annual ACM Symp. On Theory of Computing, 1990.
- [17] B. Dixon and A. K. Lenstra, "Factoring Integers Using SIMD Sieves," Advances in Cryptology - EUROCRYPT '93, 1994.
- [18] E. W. Weisstein, *The CRC Concise Encyclopedia of Mathematics*: CRC Press, 1998.
- [19] J. M. Pollard, "The Lattice Sieve," in *The Development of the Number Field Sieve*, vol. 1554, *Lecture Notes in Mathematics*: Springer-Verlag, 1991, pp. 43-49.
- [20] A. K. Lenstra and M. S. Manasse, "Factoring by Electronic Mail," Advances in Cryptology - EUROCRYPT'89, 1990.
- [21] R. D. Silverman, "The Multiple Polynomial Quadratic Sieve," *Mathematical Computing*, vol. 48, pp. 329-339, 1987.
- [22] J. Villasenor and W. H. Mangione-Smith, "Configurable Computing," *Scientific American*, vol. 276, pp. 66-71, 1997.
- [23] A. K. Lenstra, "RSA130 is Completed," <http://www.npac.syr.edu/factoring/status.html>, 1996.