

# Factoring Large Numbers with Programmable Hardware

Hea Joung Kim and William H. Mangione-Smith

{kimmer,billms}@icsl.ucla.edu

The UCLA Department of Electrical Engineering

## Abstract

The fastest known algorithms for factoring large numbers share a core sieving technique. The sieving cores find numbers that are completely factored over a prime base set raised to some power. This sieving process dominates runtime for the range of numbers that are currently of interest, and it consumes an increasing fraction of runtime for the (currently) dominant algorithms as the range of numbers increases. This report presents a hardware architecture and implementation that improves the speed of the sieving process over current implementations. The architecture is based on programmable hardware connected to parallel memory banks, and achieves improved performance through highly concurrent execution by exploiting the unique properties of prime numbers. Currently, we have a functional sieving hardware on a single field-programmable gate array that operates at 16MHz with no optimizations. At 16MHz, we are able to achieve a speedup factor of 40 over an UltraSparc Workstation. Further exploiting the primes by having 4 FPGAs operate in parallel and saturating the 10ns SRAMs will result in a speedup factor of over 250.

## 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). 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[1, 2]. 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 the fact that large integers are hard to factor. This paper presents an architecture that speeds up the sieving process 50~250 times when compared to a 200 MHz UltraSparc through parallel computing modules, pipelining within each module, and a special-purpose highly 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  $t=(p+q)/2$  and  $s=(p-q)/2$ , the  $n=((p+q)/2)^2 - ((p-q)/2)^2$  can be rewritten as  $n=t^2 - s^2$  (see Equation 1). From that equation,  $t$  can be chosen from  $n^{0.5} < t < n$ , and if a  $t^2 - n = s^2$  is a perfect square,  $n$  can be factored using  $g.c.d(t+s, n)$  or  $g.c.d(t-s, n)$ . From the algorithms of Fermat to the number field sieve (NFS) [3], the algorithms share a similar core. Most factoring algorithms quickly search for numbers  $a$  (or

$s^2$ ) that are highly composite of prime numbers raised to even powers; i.e. a sieve is applied to find these perfect square numbers.

$$\begin{aligned} x^2 - y^2 &= (x-y)(x+y) = n \\ x^2 &\equiv y^2 \pmod{n} \\ a &\equiv b^2 \pmod{n} \text{ where } a=x^2 \text{ and } b=y \end{aligned}$$

**Equation 1. Fermat's Equation**

The algorithm used for this paper is the multiple polynomial quadratic sieve (MPQS) which was the faster algorithm until the number field sieve and appears to be the most efficient for medium sized numbers (up to ~350 bits for MPQS and >350 bits for NFS – note that both share a similar sieving technique)[4, 5]. 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 about 524,338 prime numbers 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 [6]. 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

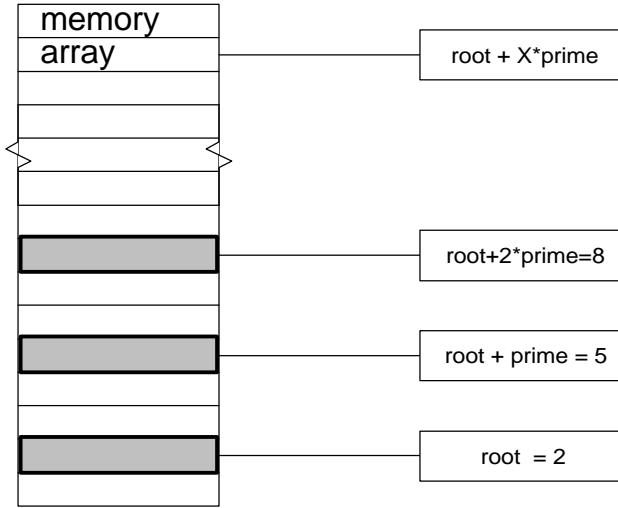
We have developed a parallel hardware sieve for a specific polynomial. The sieve working on one array is our focus. We propose an implementation that will be orders of magnitude faster than any implementation running on any general purpose computer. The goal is to find numbers that are highly composites of prime numbers as shown in Equation 2.

$$\begin{aligned} a_1 &= b_1^2 \pmod{n} = p_1^{a11} * p_2^{a12} * p_3^{a13} * p_4^{a14} * \dots * p_i^{a1j} \\ a_2 &= b_2^2 \pmod{n} = p_1^{a21} * p_2^{a22} * p_3^{a23} * p_4^{a24} * \dots * p_i^{a2j} \\ a_3 &= b_3^2 \pmod{n} = p_1^{a31} * p_2^{a32} * p_3^{a33} * p_4^{a34} * \dots * p_i^{a3j} \\ &\vdots \\ a_k &= b_k^2 \pmod{n} = p_1^{ak1} * p_2^{ak2} * p_3^{ak3} * p_4^{ak4} * \dots * p_i^{akj} \end{aligned}$$

**Equation 2. Composites of Prime numbers**

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  $i$ begin,  $i$ end, sieve array size, and the logarithm of the prime. The  $i$ begin value is used to select the initial roots and prime. The roots are representative of other large integers

that need to be sieved. Using the multiple primes and roots, the entire sieve array is traversed iteratively as shown in Figure 1. 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 for each prime, the process stops and continues with a new root and prime.



**Figure 1. Memory array for Sieving with root=2 and prime=3**

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 siever 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 [7, 8], 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 an interface to an i960 PCI board. The platform contains five Xilinx XC4085XL-3.

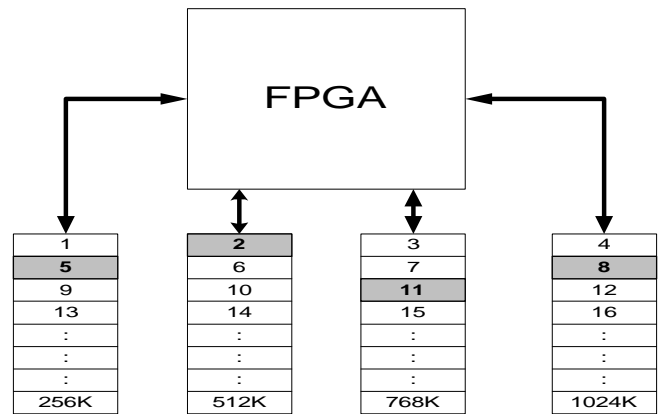
The board provides four computing nodes – 4 FPGAs. The computing nodes are configured through the fifth FPGA, which also manages the interface bus and all external IO. A 96-signal programmable ring-bus connects each of the computing FPGAs. Each computing node has four banks of local SRAM that are 16-bits 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 Cyclone Microsystems board 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.

#### 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 nature of prime numbers. Figure 2 shows an example, where the prime being used is three.

#### 4 Memory Banks per FPGA



**Figure 2. 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. 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 16 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.

Execution times on various workstation for 85,200 sieveops are listed in Table 1. The performance is for a low sieve with iend-ibeg=55 using a sieve length of  $2^{20}=1,048,576$ .

Machine (CPU)	Time
Sparc5 (microSparcII@110MHz)	110ms
Sparc20 (SuperSparcII@75MHz)	80ms
Ultra2 (UltraSparc@200MHz)	27.9ms

Table 1. Time to Execute ~85,200 Sieveops

A single FPGA on the Mojave board performs 85,200 sieveops in 2.66ms. The layout for the circuit used shows that 20% or 634 out of 3136 CLB's are being utilized. 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 70ns access parts that we are currently using. An FPGA can perform 4 sieve operations (4 addresses per clock) in 2 clock cycles (read, add+write). Therefore, at 16MHz,  $(85,200 * \frac{1}{4} \text{ parallel} * 2 \text{ cycles} * 62.5\text{ns}) = 2.66\text{ms}$  per sieveop. Achievable performances at higher frequencies are list in Table 2.

MHz	# cyc	parallel	sieveops	time (ms)
16	2	4	85000	2.66
33	2	4	85000	1.29
100	2	4	85000	0.43
133	2	4	85000	0.32

Table 2. Time to perform 85,000 sieve operations on a Single FPGA

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 as shown in Figure 3. 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.

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 (~0.43ms @100MHz). For the low prime sieve, we have a factor of 4 further improvement by using a 5 FPGA board with 4 processing elements (sieving time ~0.1075ms).

### 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

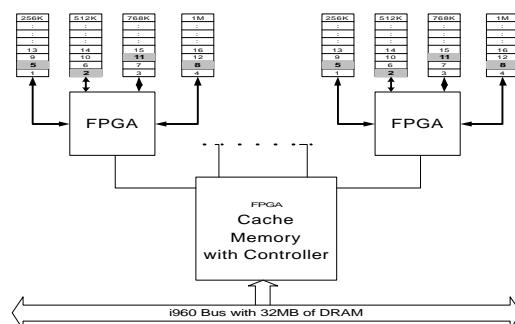


Figure 3. Parallel Processing using 4 FPGAs

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 is limited by the access to the SRAM, it will not be necessary to consider an ASIC implementation. In sieving operations, the FPGA is essentially performing as well as any possible ASIC.

## 6 Conclusion and Future Work

The hardware implementation provides further speed up in the number factoring algorithm. Although we claim a speed up of over 250 when using 16 SRAMs in parallel with 4 FPGAs, the speedup can be greater with more dedicated system. Given that current 4085s have 448 I/O pins, there could be 14 SRAMs per FPGA. With this type of implementation, the speedup will be even greater.

Furthermore, this hardware sieving with Number Field Sieve will factor the RSA-129 in less time it took to factor RSA-129 using MPQS [9]. RSA130 was factored using 10% of compute time used to factor RSA129 using NFS.

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. Most importantly, the implementation can be scaled to execute sieves in a highly parallel architecture due the nature of prime numbers.

## 7 Reference

- [1] R. L. Rivest, A. Shamir, and L. Adleman, "On digital signatures and public-key cryptosystems," presented at 1977 IEEE International Symposium on Information Theory, Ithaca, NY, USA, 1977.
- [2] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, pp. 120-6, 1978.
- [3] A. K. Lenstra, J. H. W. Lenstra, M. S. Manasse, and J. M. Pollard, "The Number Field Sieve," presented at Proc. 22nd Annual ACM symp. On Theory of Computing, 1990.

- [4] R. D. Silverman, "The multiple polynomial quadratic sieve," in *Mathematics of Computation*, vol. 177, 1987, pp. 329-39.
- [5] A. K. Lenstra, H. W. Lenstra, Jr., M. S. Manasse, and J. M. Pollard, "The number field sieve," presented at 1990 Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing, Baltimore, MD, 1990.
- [6] A. K. Lenstra and M. S. Manasse, "Factoring by Electronic Mail," presented at Advances in Cryptology - EUROCRYPT'89, 1990.
- [7] 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. 364-372, 1998.
- [8] J. Villasenor and W. H. Mangione-Smith, "Configurable computing," *Scientific American*, vol. 276, pp. 54-9, 1997.
- [9] A. K. Lenstra, "RSA130 is Completed," : <http://www.npac.syr.edu/factoring/status.html>, 1996.