

# A Scalable Reconfigurable Architecture For Divisibility Testing Of Variable Long Precision Numbers

Easwaran Raman  
Indian Institute of Science,  
Bangalore, India.  
easwar@csa.iisc.ernet.in

Lakshmi N Chakrapani  
Georgia Institute of  
Technology, Atlanta, USA.  
nsimhan@cc.gatech.edu

Karthik Sankaranarayanan  
University of Virginia,  
Charlottesville, USA.  
karthick@cs.virginia.edu

Ranjani Parthasarathi  
Anna University,  
Chennai, India.  
[rp@annauniv.edu](mailto:rp@annauniv.edu)

## Abstract

*Divisibility testing plays an important role in areas like cryptography and a fast hardware has many applications. This paper discusses an algorithm for divisibility testing of Variable Long Precision (VLP) numbers and its implementation on a reconfigurable target. The algorithm exhibits a high degree of parallelism and scalability and hence helps in deciding a suitable cost-performance tradeoff. The performance of the algorithm is analyzed and the performance of the hardware implementation is compared with that of the GNU multi-precision library.*

## 1. Introduction

Variable Long Precision Arithmetic (VLPA) techniques are used to perform arithmetic operations on large numbers without loss of precision. Several algorithms exist to perform VLP arithmetic operations, which find applications in numerous areas like computational algebra and elliptic curve cryptography [5]. This paper introduces an algorithm for fast divisibility testing of VLP numbers. This algorithm is characterized by the fact that it adopts a generic approach for divisibility testing of any pair of numbers. The algorithm exhibits massive amount of parallelism and scalability enabling the designer to trade cost for performance.

## 2. The Ekadhika Algorithm

The algorithm for divisibility testing takes a positive integer radix  $x$  and two normalized Variable Long Precision (VLP) integers (the dividend and the divisor) as its inputs. The algorithm then decides if the dividend is divisible by the divisor. The algorithm has three major operations - finding the Ekadhika (a special multiplier), an operation called osculation (a process of repetitive multiply-add operations which reduces the problem to an easily solvable, smaller divisibility-testing problem.) and a smaller divisibility test.

The Ekadhika is a special multiplier derived from the divisor. Let the radix of the numbers be  $x$ , the divisor be  $A$  and the dividend be  $B$ . Let least significant digit of  $A$  be  $a_0$  and the rest of the digits of  $A$  constitute the number  $a$  i.e.,

$$A = a*x + a_0 \quad (1)$$

Let  $y$  be such that

$$a_0 * y = x-1 \pmod{x} \quad (2)$$

Such a  $y$  cannot be found if the divisor is a multiple of  $r$ , where  $r$  divides  $x$ . In that case, the dividend and the divisor are repeatedly divided by  $r$  such that the divisor is no more a multiple of  $r$ . In the process, if the dividend is not divisible by  $r$ , but the divisor is, then obviously the divisibility test fails. While implementing the algorithm, since the radix is typically

a power of 2,  $r$  is also a power of 2 and the repeated division is reduced to inexpensive shift operations. The Ekadhika  $E$  is given by

$$E = (A * y + 1) / x \quad (3)$$

The division by  $x$  is achieved by shifting. The process of osculation, which involves many iterations of multiply-add operations, is the next step of the algorithm. Osculation is performed as follows. For any osculation iteration, let  $M$  be the multiplicand' for that iteration, whose least significant digit is  $m_0$ . Let  $m$  be the number formed by the rest of its digits i.e.,

$$M = m * x + m_0 \quad (4)$$

Initially set  $M=B$ . During every iteration of the osculation process, the Ekadhika  $E$  is multiplied by  $m_0$  and to this the value obtained by dividing  $M$  by  $x$  is added. Let the number that results due to the multiply-add operation be  $C$ . Notationally,

$$C = m_0 * E + m \quad (5)$$

$C$  is typically smaller than the multiplicand itself, and is the new multiplicand for the next iteration (i.e. for the next iteration set  $M=C$ ). This process of osculation is continued until  $C$  has one digit more than that of the Ekadhika.

Once  $C$  becomes small enough (i.e., with one digit more than the divisor at most), it is tested for divisibility by the divisor using Knuth's guessing technique [4] using a single vector multiplication. The result of the smaller divisibility test indicates the result of the original divisibility test.

## 3. Proof of correctness

Since the algorithm converts the original divisibility test to a smaller one, to prove its correctness, it is sufficient to prove the equivalence between the results of the original and the smaller divisibility tests. Using (3) in (5), it can be seen that

$$C * x = m_0 * A * y + M \quad (6)$$

If  $C$  is divisible by  $A$ , clearly  $M$  is. On the other hand, if  $M$  is divisible by  $A$ ,  $M$  can be expressed as  $A*k$  for some non-negative integer  $k$ . Substituting  $(A*k-m*x)$  for  $m_0$  in (5) and using (3),  $C$  can be expressed as

$$C = A (k*E - m*y) \quad (7)$$

That is, if  $M$  is divisible by  $A$ , so is  $C$ . Thus, the equivalence is proved.

However, this is not sufficient for the correctness of the algorithm because, as a result of osculation iteration, the number might get repeated i.e.,  $C$  might equal  $M$ . In such a

case, the problem is not reduced to a smaller problem. So, one should resort to some other means to test the divisibility. However, this repetition can happen only when the divisibility

problem has already been made small enough that Knuth's guessing can be used. If  $C$  remains the same in two successive iterations,  $M=C$  in (6) i.e.,

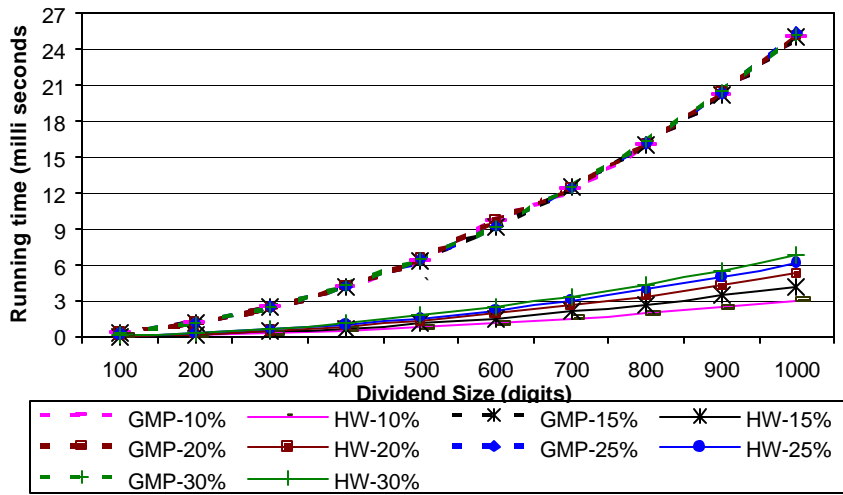


Figure 1 – Running time comparison – GMP and Hardware implementation

$$M = (m_0 * y / (x-1)) * A \quad (8)$$

Here,  $m_0$ ,  $y$ ,  $x-1$  are all single digit numbers and hence  $(m_0 * y / (x-1))$  can have at most one digit before the decimal place. Thus  $M$  can have at most one digit more than  $A$ . Thus, as stated above, the problem is small enough.

### 3. Implementation and Performance analysis

The implementation is in the form of a host + reconfigurable co-processor architecture. The host is provided with the divisor and the dividend. The host calculates the Ekadhika of the divisor and loads it into the co-processor. The co-processor performs the osculation and indicates when the osculation process is complete. The host then performs a single check using the Knuth guessing technique and declares if the dividend is divisible by the divisor. For the validation of the design of the hardware, an 8-bit version of the algorithm has been implemented on a Xilinx 4010XL FPGA on a XESS 40xv board. The unit operates at 13.9 MHz and is interfaced to the host via a parallel port. In this implementation, the communication costs for transferring the numbers to the co-processor has not been considered. A faster interface such as through the PCI bus would imply lesser communication costs. A 'C' program running on the host configures the FPGA, performs the preprocessing operations like calculation of the Ekadhika, loads the numbers into the co-processor and initiates the calculation. When the co-processor terminates on the input, the program reads the output from the co-processor and performs the Knuth's guessing algorithm.

For analyzing performance of the hardware, an implementation of the multiplier unit on a Xilinx Virtex device is considered. In our timing simulations, the 3-stage pipelined multiplier operates with a frequency of 31MHz. With this as the operating frequency of the divisibility-testing unit, the hardware implementation is compared with a library software implementation that uses the GNU Multi-Precision

(GMP). The software implementation was run on a Sun UltraSPARC II 350 MHz machine. It checks for divisibility by looking at the remainder of actual division. Figure 1 shows the execution times of the hardware and the software implementations for different values of dividend and divisor sizes. It is to be noted that the execution time of the hardware does not include the time taken for pre and post processing. The table lookup for Ekadhika and the quotient guess take constant time. Ekadhika computation and verifying the guessed quotient take at most 3 vector multiplications. Since these values are very small compared to the total time, they are ignored.

In the figure, various lines in each category show different ratios of the divisor size to the dividend size. For example, the 'HW-25%' line shows the execution time of the hardware for the size of the divisor being one-fourth the size of the dividend

### References

- [1] JSS Bharathi Krishna Tirthaji Maharaj "Vedic Mathematics", Motilal Banarasi Das, Varanasi, 1986.
- [2] Hwang, Kai. "Computer Arithmetic -Principles, Architecture and Design", John Wiley & Sons Inc. 1979.
- [3] Jhunjhunwala, A. "Indian Mathematics - an Introduction", Wiley Eastern Limited, Madras, 1995.
- [4] Knuth, D.E. "The Art of Computer Programming", Vol.2 (Semi-numerical algorithms) Addison Wesley, 1980.
- [5] Tenca, A.F. and Ercegovac, M.D. "A Variable Long-Precision Arithmetic Unit Design for Reconfigurable Coprocessor Architectures". In Proc. IEEE Symposium on Field-Programmable Custom Computing Machines, April 1998.