Very Fast and Exact Accumulation of Products

(Required by the IEEE Standards Committee P1788)

Ulrich Kulisch

1 Introduction

The IFIP Working Group on Numerical Software and other scientists repeatedly requested that a future arithmetic standard should consider and specify an exact dot product (EDP) [1,2]. On Nov. 18, 2009 the IEEE standards committee P1788 on interval arithmetic accepted a motion [3] for including the EDP into a future interval arithmetic standard. In Numerical Analysis the dot product is ubiquitous. It is not merely a fundamental operation in all vector and matrix spaces. It is the EDP which makes residual correction effective. This has a direct and positive influence on all iterative solvers of systems of equations. The EDP is essential for fast long real and long interval arithmetic, as well as for assessing and managing uncertainty in computing. By operator overloading variable precision interval arithmetic is very easy to use. With it the result of every arithmetic expression can be guaranteed to a number of correct digits.

Actually the simplest and fastest way for computing a dot product is to compute it exactly. By pipelining, it can be computed in the time the processor needs to read the data, i.e., it comes with utmost speed. By a sample illustration the poster informally specifies the implementation of the EDP on computers. While [3] defines **what** has to be provided, how to embed the EDP into the new standard IEEE 754, and how exceptions like NaN are to be dealt with, the poster illustrates **how** the EDP can be implemented on computers. There is indeed no simpler way of accumulating a dot product. Any method that just computes an approximation also has to consider the relative values of the summands. This results in a more complicated method. The hardware needed for the EDP is comparable to that for a fast multiplier by an adder tree, accepted years ago and now standard technology in every modern processor. The EDP brings the same speedup for accumulations at comparable costs.

2 Informal Description for Realizing an Exact Dot Product

Let $a = (a_i)$, $b = (b_i)$ be two vectors with n components which are floating-point numbers $a_i, b_i \in \mathbb{F}(b, l, e1, e2)$, for $i = 1(1)n$. We compute the sum

$$s := \sum_{i=1}^{n} a_i \cdot b_i = a_1 \cdot b_1 + a_2 \cdot b_2 + \ldots + a_n \cdot b_n, \qquad a_i \cdot b_i \in \mathbb{F}(b, 2 \cdot l, 2 \cdot e1, 2 \cdot e2), \text{ for } i = 1(1)n,$$

where all additions and multiplications are the operations for real numbers.

All summands can be taken into a fixed-point register of length $2 \cdot e2 + 2 \cdot l + 2 \cdot |e1|$ without loss of information.

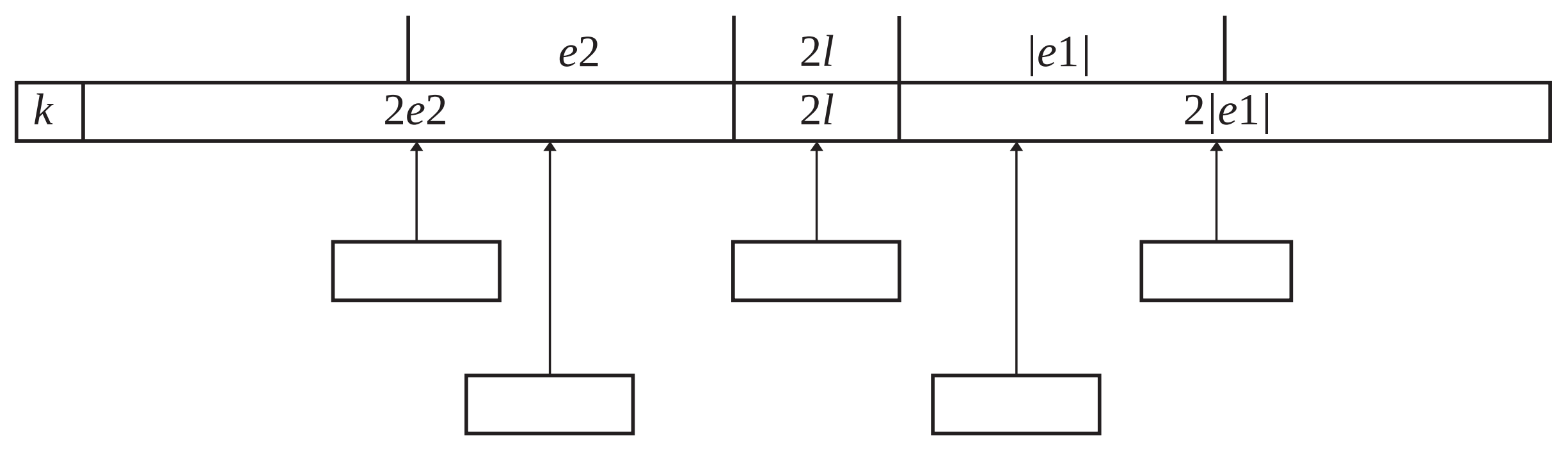


Figure 1 Complete register for exact scalar product accumulation.

If the register is built as an accumulator with an adder, all summands could be added in without loss of information. To accommodate possible overflows, it is convenient to provide a few, say k, more digits of base b on the left. k can be chosen such that no overflows will occur in the lifetime of the computer.

For IEEE-arithmetic double precision we have:
$b = 2$; 64 bits word length; 1 bit sign; 11 bits exponent; $l = 53$ bits; $e1 = -1022$, $e2 = 1023$. With $k = 92$ the entire unit consists of
$L = k + 2 \cdot e2 + 2 \cdot l + 2 \cdot |e1| = k + 4196$ bits $= 4288$ bits. It can be represented by 67 words of 64 bits. L is independent of n.

The following figure gives an informal description for realizing an EDP. The long register (here represented as a chest of drawers) is organized in words of 64 bits. The exponent of the products consists of 12 bits. The leading 6 bits give the address of the three consecutive drawers to which the summand of 106 bits is added. The low end 6 bits of the exponent are used for the correct positioning of the summand within the selected drawers. A possible carry is absorbed by the next more significant word in which not all bits are 1. For fast detection of this word a flag is attached to each word. It is set 1 if all bits of the word are 1. This means that a carry will propagate through the entire word. In the figure the flag is shown as a red point. As soon as the exponent of the summand is available the flags allow selecting and incrementing the carry word. This can be done simultaneously with adding the summand into the selected drawers.

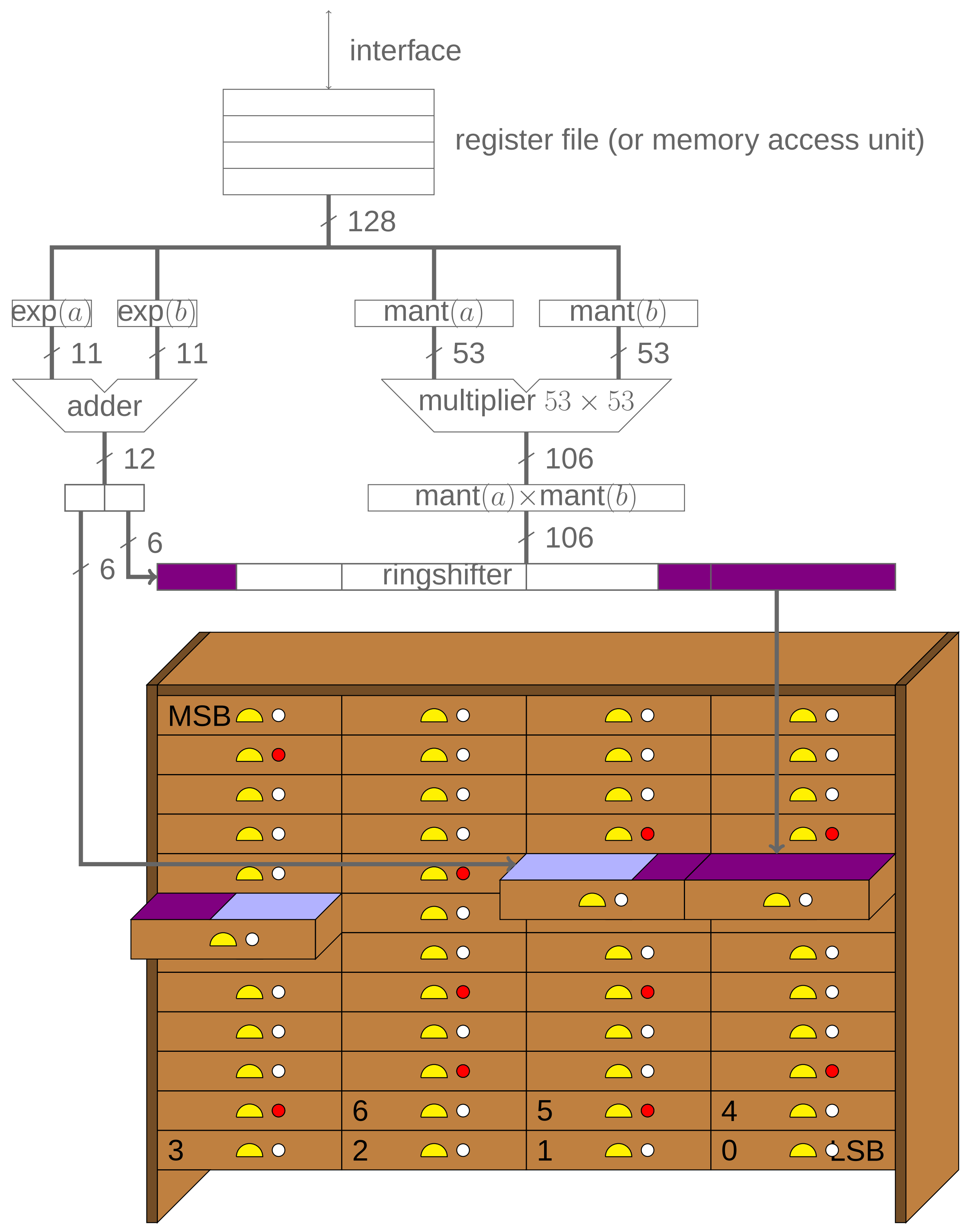
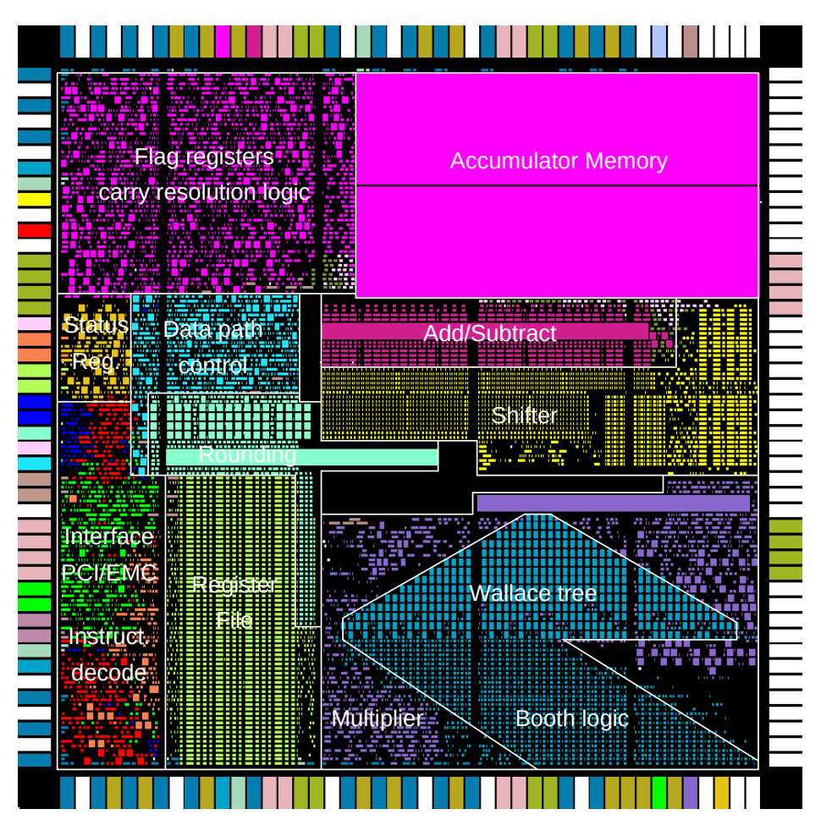


Figure 2 Mechanical computing devices equipped with the desired capability:
Burkhart Arithmometer, Glashütte, Germany, 1878; **Brunsviga**, Braunschweig, Germany, 1917; **MADAS**, Zürich, Switzerland, 1936; **MONROE**, New Jersey, USA, 1956.

3 Circuitry for the Exact Dot Product



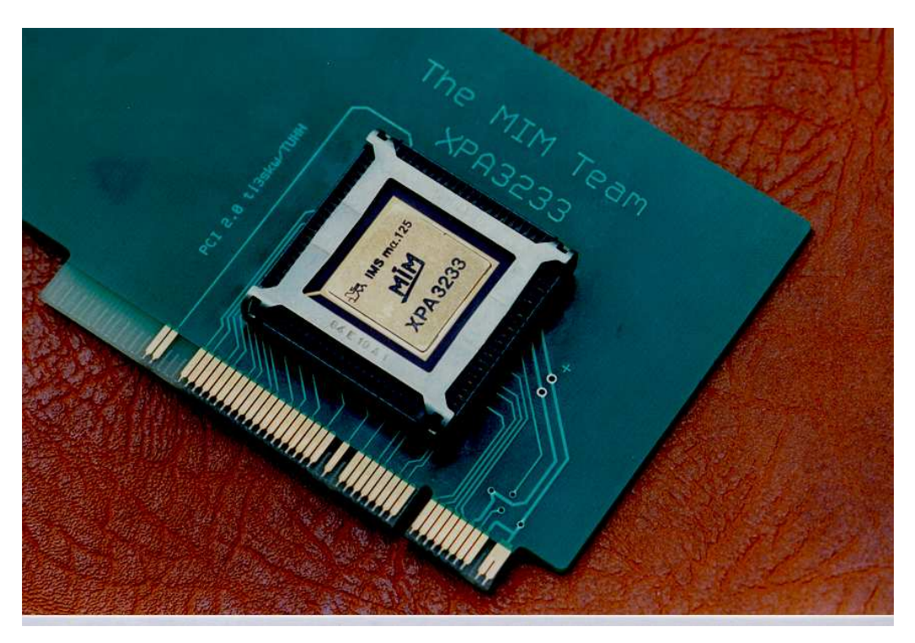


Figure 3 Functional units, chip and board of the vector arithmetic coprocessor XPA 3233.

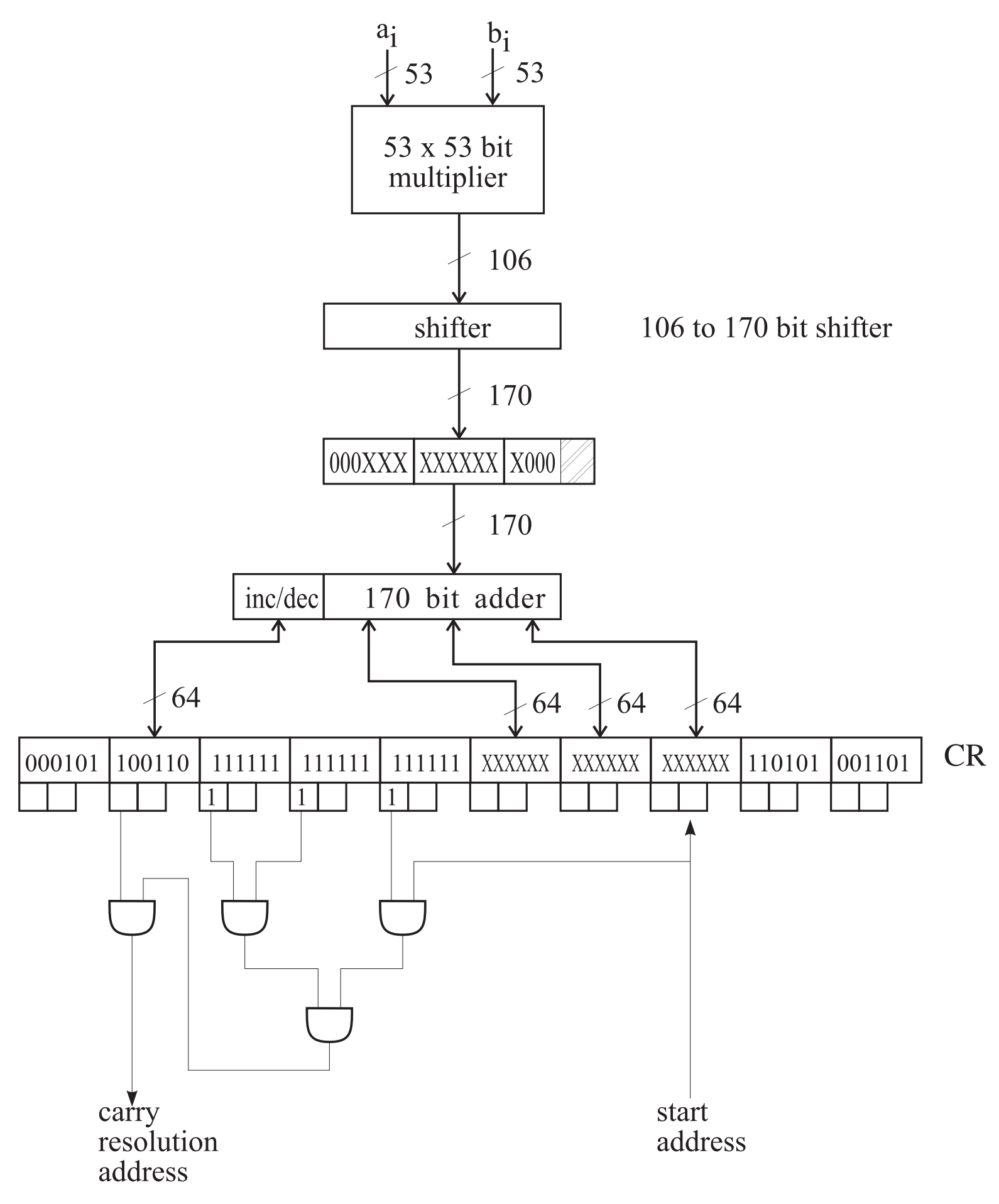


Figure 4 Parallel accumulation of a product into the CR.

4 Two simple applications

To be successful interval arithmetic has to be complemented by some easy way to use multiple or variable precision arithmetic. The fast and exact dot product is the tool to provide this very easily.

With the EDP quadruple or other multiple precision arithmetic can easily be provided on the computer. This enables the use of higher precision operations in numerically critical parts of a computation. It helps to increase software reliability. A multiple precision number is represented as an array of floating-point numbers. The value of this number is the sum of its components. The number can be represented in the long register in the arithmetic unit. See the other column. Addition and subtraction of multiple precision numbers can easily be performed in this register. Multiplication of two such numbers is simply a sum of products. It can be computed easily and fast by means of the EDP. For instance, using fourfold precision the product of two such numbers $a = (a_1 + a_2 + a_3 + a_4)$ and $b = (b_1 + b_2 + b_3 + b_4)$ is obtained by

$$\begin{aligned} a \cdot b &= (a_1 + a_2 + a_3 + a_4) \cdot (b_1 + b_2 + b_3 + b_4) \\ &= a_1 b_1 + a_1 b_2 + a_1 b_3 + a_1 b_4 + a_2 b_1 + \ldots + a_4 b_3 + a_4 b_4 \\ &= \sum_{i=1}^{4} \sum_{j=1}^{4} a_i b_j. \end{aligned}$$

The result is a sum of products of floating-point numbers. It is independent of the sequence in which the summands are added.

A very impressive application is considered in [5], an iteration with the logistic equation (dynamical system)

$$x_{n+1} := 3.75 \cdot x_n \cdot (1 - x_n), \quad n \geq 0.$$

For the initial value $x_0 = 0.5$ the system shows chaotic behavior.

Double precision floating-point or interval arithmetic totally fail (no correct digit) after 30 iterations while long interval arithmetic still computes correct digits of a guaranteed enclosure after 2790 iterations.

References

[1] The *IFIP WG - IEEE 754R letter*, dated September 4, 2007.

[2] The *IFIP WG - IEEE P1788 letter*, dated September 9, 2009.

[3] U. Kulisch, V. Snyder: *The Exact Dot Product as Basic Tool for Long Interval Arithmetic*.

[4] U. Kulisch: *Computer Arithmetic and Validity – Theory, Implementation, and Applications*, de Gruyter, Berlin, New York, 2008.

[5] F. Blomquist, W. Hofschuster, W. Krämer: *A Modified Staggered Correction Arithmetic with Enhanced Accuracy and Very Wide Exponent Range*. In: A. Cuyt et al. (eds.): *Numerical Validation in Current Hardware Architectures*, Lecture Notes in Computer Science LNCS, vol. 5492, Springer-Verlag Berlin Heidelberg, 41-67, 2009.

Institut für Angewandte und Numerische Mathematik • KIT Karlsruhe • D-76128 Karlsruhe GERMANY
Ulrich.Kulisch@kit.edu