On the construction of very large integer multipliers

G. Hotz
Fachbereich Informatik
Universität des Saarlandes
D-W–6600 Saarbrücken

P. Molitor
Fachbereich Informatik
Universität des Saarlandes
D-W–6600 Saarbrücken

W. Zimmer
Forschungszentrum Informatik
Haid-und -Neu-Str. 10-14
D-W–7500 Karlsruhe

Abstract

In this paper we present a fast shared multiplier for very large numbers. Realizing this concept for any input length (e.g., 512, 1024 or 2048 bits) only needs three types of chips.

1 Introduction

There are various applications of fast multipliers for large integers.

One of the most interesting applications is the following one. The most important method for public-key systems [7,9] is the RSA-method [10], whose security is based on the high complexity of computing the primes of large integers. Increasing the length of the used numbers improves its security. Using fast multipliers for those numbers allows to do this nearly without loss of running time of the encryption and decryption process. On the other hand, some of today's fastest factorization algorithms (e.g., factorization with elliptic curves [8]) also use multiplications of large numbers. This fact just slightly deteriorates the system's security.

Wallace-tree multipliers have optimal running time $c \cdot \log n$ where c is a small constant and n is the input length. Unfortunately, they can only be realized up to a certain length on one single chip (in the examples of this paper we assume that the maximum possible length is 32), so that nowadays large multiplications are simulated by software or micro-programs. The goal of our work is to speed up long integer multiplications by constructing a multichip multiplier. For this, we generalize the already mentioned Wallace-method.

The design goals are the following:

- *Optimal running time* $c \cdot \log n$ where c is a small constant. In this connection, the main problem is to reduce the amount of (time-consuming) communication between the chips of the shared circuit.
- *Small number of different types of chips* in order to achieve low development costs.
- *Regularity of the wiring* in order to reduce the complexity of the circuit because of the enormous amount of wire segments between and on the chips.
- *Easy testability* of the used chips so that one can rely on the correctness of the circuit.
- The *amount of chips should be kept low* in order to realize the shared multiplier by using only few platines.

The multiplier presented in this paper is an optimal-time multiplier with a very small constant in the $O(\log n)$-term which results from an efficient carry handling during the reduction of the multiplication matrix. It uses three different types of chips. Two of them have to be designed, the other one is a standard 32-bit multiplier. There are parametrized layouts of the used chips (the parameters determine the input length) (see [2,4]), whose testability can be improved by small extensions [1,3,6].

Pipelining this multiplier is a further possibility to speed up applications which use a lot of large multiplications which are independent of each other. This requires a new chip and some extensions on the other ones (see [11] for more details).

TH0367-3/91/0000/0266$01.00 © 1991 IEEE

2 The structure of the n-bit multiplier

The first idea of the construction of very large multipliers is to partition the whole circuit into the following three stages (see figure 1).

In the first stage, 32-bit blocks of the input numbers A and B are multiplied according to multiplications of numbers on base 2^{32}. The input of the second stage consists of the results of the first stage. This large amount of 64-bit integers (of different valences) are reduced to two $2n$-bit numbers according to the Wallace-method. The third stage is a fast shared $2n$-bit adder which computes the final result of the multiplication of A and B.

The partition in three independent stages causes a considerable reduction of the design complexity. The stages can be separately designed. It is possible to replace one stage without affecting the other ones.

2.1 Stage 1 - Partition of the multiplication matrix

We separate the two input numbers $A = \alpha_{n-1} \dots \alpha_0$, $B = \beta_{n-1} \dots \beta_0$ in blocks of length b_1 which is the maximum input length of a Wallace-tree multiplier realizable on one single chip (e.g., $b_1 = 32$). Figure 2 and the corresponding equation illustrate this partition of the multiplication matrix.

2.2 Stage 2 - Reduction of the multiplication matrix

The input of the second stage consists of the large amount of the results of the multiplications of the first stage. For an input length $n = 1024$, the first stage generates 65536 output bits if 32-bit multipliers are used, which have to be reduced in the second stage.

The 'structure of the input' of stage 2 ($s_1 = 4$) is shown in the left part of figure 3. We have arranged the $A_i B_j$'s of stage 1 according to their valences. (More formally, let $a = (a_{m-1} \dots a_0) \cdot 2^p$ be a binary number, i.e., a is equivalent to the binary number $(a_{m-1} \dots a_0)$ which is p times leftshifted, we denote p by *valence of* a.)

The input of stage 2 is split into columns of width b_2 (see right part of figure 3) so that the i-th column processes a certain amount h (height of the column) of b_2-bit numbers of valence $i \cdot b_2$. The input of a column is distributed on several chips of type "chip 2a" which are arranged in a tree-like structure which we denote by *reduction-tree*.

Each chip 2a consists of a Wallace-tree whose size is limited by the maximum number of pins available on a chip (nowadays about 256 pins) and not by the used area, i.e., the maximum number of pins of a chip is the bottle-neck in stage 2.

In a Wallace-tree, two different steps alternate with one another. The first step is a $3to2$-reduction by fulladders, the second one is the shifting of the carry bit of the fulladders to the next higher position.

The carries between neighbouring columns make the difficulties of stage 2. The shifting of these carries from the chips of the i-th column to the chips of the $i+1$-th column causes a lot of (relatively slow) communication between chips.

In order to get an efficient carry handling, our idea is to enlarge the bitlength of the Wallace-tree (of chip 2a) from the column-width b_2 to $b_2 + k$. Thereby, all the carries which previously had to be shifted to the next column now accumulate in the first k positions (k accordingly chosen).

So, the output of each reduction tree consists of two (b_2+k)-bit numbers although the column-width is only b_2. The first k positions of both numbers in the i-th column are given to the next higher column, just now, and the carry handling is finished by reducing the two shifted k-bit numbers of the i-th column and the two corresponding b_2-bit numbers of the $i+1$-th column to two b_2-bit numbers. This last step is realized by "chip 2b".

Chip 2b consists of a Wallace-tree (built up of parallel $4to2$-reduction cells) for four b_2-bit numbers. As two of the four input numbers are significantly smaller than the other both ($k << b_2$), this reduction can be realized without any communication between chips, i.e., there is no carry any more.

This efficient carry handling results in a considerable decrease of the communication between chips. The complexity of the circuit is reduced because of the independence of the columns and the simple carry handling. Moreover, chip 2a and chip 2b can be integrated into one chip by some small extensions.

2.3 Stage 3 - Final addition

We use the idea of conditional-sum-adders for the shared circuit realizing the final $2n$-bit addition. The chips themselves of this stage are modified carry-look-ahead adders (CLA-adder).

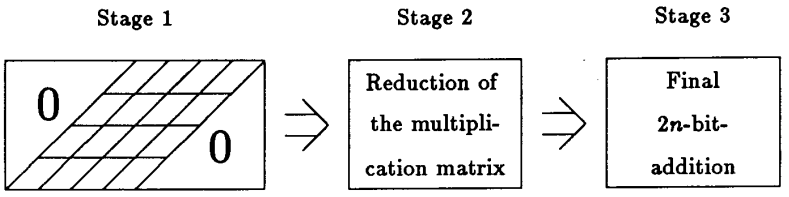
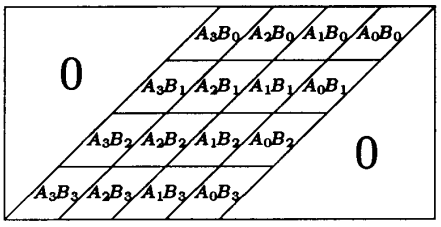


Figure 1: Schematic structure of the $2n$-bit-multiplier



$$
\begin{aligned}
A \cdot B &= (A_{s_1-1} \dots A_0) \cdot (B_{s_1-1} \dots B_0) \\
&= \sum_{i,j \in \{0,\dots,s_1-1\}} A_i \cdot B_j \cdot 2^{(i+j)b_1}
\end{aligned}
$$

$$
b_1 = 32 \qquad s_1 = n/b_1
$$

$$
|A_i| = |B_i| = b_1 \qquad \forall i \in \{0, \dots, s_1 - 1\}
$$

Figure 2: Partition of the multiplication matrix and corresponding equation

As in stage 1, we separate the two $2n$-bit numbers X, Y in s_3 blocks $D_i = (X_i, Y_i)$ of length b_3, which is the maximum input length of a fast adder realizable on one single chip. A modified CLA-adder, which we denote by 'chip 3a', is assigned to each block. It computes the values $X_i + Y_i$, $X_i + Y_i + 1$ and the information, whether block D_i generates/propagates an (incoming) carry.

Chip 3b which is only needed once is given by an usual carry-computation part of a CLA-adder. Using the propagating/generating information of the blocks (see chip 3a) chip 3b computes the carries between the blocks, i.e., the output of chip 3b consists of the incoming carry for block D_i $\forall i \in \{0, 1, \dots, s_3 - 1\}$. Dependent on this information, multiplexers integrated on chip 3a select the right result ($X_i + Y_i$ or $X_i + Y_i + 1$) (see figure 4).

The two circuits chip 3a and chip 3b can be integrated on one chip using multiplexers because of their similar structure.

3 Summary

We have presented an optimal-time multiplier for very large numbers. It consists only of three different, easily testable types of chips (see [1,3]). One of them is a 32-bit multiplier which has not to be developed any more. We expect a speed up of factor 100-1000 (dependent on the input length) in comparison to a simulation by software. A further speed up of applications using many independent multiplications can be attained by pipelining the multiplier. More details on these extensions can be found in [11].

The 1024-bit multiplier can be realized using only four platines at a rough estimate. The multiplier itself is not yet realized.

Acknowledgements

This work was supported by *Deutsche Forschungsgemeinschaft* under contract SFB 124 *VLSI Entwurfsmethoden and Parallelität* TP B1.

References

[1] B. Becker. Efficient testing of optimal-time adders. *IEEE Transactions on Computers*, C-37:1113–1121, 1988.

[2] B. Becker, Th. Burch, G. Hotz, D. Kiel, R. Kolla, P. Molitor, H. G. Osthof, G. Pitsch, and U. Sparmann. A graphical system for hierarchical specifications and checkups of VLSI circuits. In *Proceedings of the 1st European Design Automation Conference (EDAC90)*, pages 174–179, 1990.

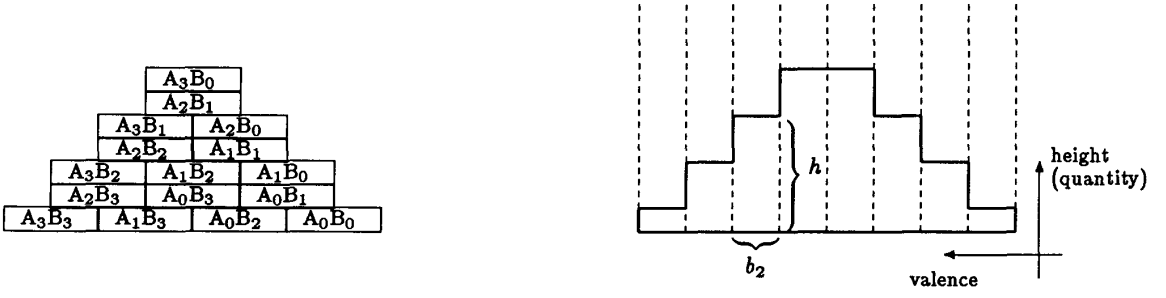


Figure 3: Schematical structure of the input of stage 2 and its partition in columns

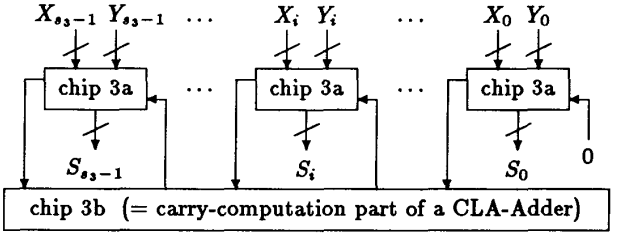


Figure 4: Distribution of the addition on several chips

[3] B. Becker and J. Hartmann. Optimal-time multipliers and c-testability. In *Proceedings of the 2nd Annual Symposium on Parallel Algorithms and Architectures*, 1990.

[4] B. Becker, G. Hotz, R. Kolla, P. Molitor, and H.G. Osthof. Hierarchical design based on a calculus of nets. In *Proceedings of the 24th Design Automation Conference (DAC87)*, pages 649–653, June 1987.

[5] B. Becker and R. Kolla. On the construction of optimal time adders. *Fundamenta Informaticae, Annales Societatis Mathematicae Polonae*, XII:205–220, 1989.

[6] B. Becker and U. Sparmann. Regular structures and testing: RCC-adders. In *Proceedings of the 3rd Aegean Workshop on Computing*, pages 288–300, 1988.

[7] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE-IT*, 22:644–654, 1976.

[8] Neal Koblitz. *A Course in Number Theory and Cryptography*. Springer Verlag, 1987.

[9] R.L. Rivest. *Handbook of theoretical computer science*, chapter 13. Volume A, Elsevier Science Publishers B.V., 1990.

[10] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *CACM*, 21:120–126, 1978.

[11] W. Zimmer. *Realisierung eines schnellen 1024-Bit Multiplizierers*. Master's thesis, Universität des Saarlandes, Saarbrücken, 1990. 113 Seiten.