Journal of Universal Computer Science, vol. 12, no. 5 (2006), 499-511
submitted: 26/8/05, accepted: 24/5/06, appeared: 28/5/06

Reversible Karatsuba's Algorithm

L. A. B. Kowada
Universidade Federal do Rio de Janeiro - UFRJ
Rio de Janeiro, RJ, 20550-900, Brazil
kowada@cos.ufrj.br

R. Portugal
Laboratório Nacional de Computação Científica - LNCC
Getúlio Vargas 333, Petrópolis, RJ, 25651-075, Brazil
portugal@lncc.br

C. M. H. Figueiredo
Universidade Federal do Rio de Janeiro - UFRJ
Rio de Janeiro, RJ, 20550-900, Brazil
celina@cos.ufrj.br

Abstract: Karatsuba discovered the first algorithm that accomplishes multiprecision integer multiplication with complexity below that of the grade-school method. This algorithm is implemented nowadays in computer algebra systems using irreversible logic. In this paper we describe reversible circuits for the Karatsuba's algorithm and analyze their computational complexity. We discuss garbage disposal methods and compare with the well known Bennett's schemes. These circuits can be used in reversible computers which have the advantage of being very efficient in terms of energy consumption. The algorithm can also be used in quantum computers and is an improvement of previous circuits for the same purpose described in the literature.
Key Words: Reversible computing, reversible circuit, arithmetic operations, multiplier
Category: B.2, B.6, F.1, F.2

1 Introduction

Parallel versions of the Karatsuba's multiplication algorithm [12] have been analyzed in the last years [7, 11, 5]. All these versions are to be implemented in irreversible devices. It is known that Moore's law is becoming weaker and will break down in the most pessimistic forecast in the year 2020. It is interesting, at least from the theoretical point of view, to analyze alternative versions of Karatsuba's algorithm which could be implemented in reversible devices. Such devices do not suffer many drawbacks of miniaturization of computer components, such as heat production. Besides, they can be very efficient in terms of power usage. In principle they can perform calculation with zero energy expenditure. Such devices may be useful in extreme situations such as deep space exploration.

Reversible computing is a growing area of Computer Science which provides a way to go beyond the physical limits to computation given by the Landauer limit [13] of $k\mathbb{T}\ln 2$ energy dissipated per bit operation, where k is the Boltzmann's constant and $\mathbb{T}$ is the temperature of the computer environment. The initial motivation to endeavor in this area was the desire to match the principles of computation with the principles of physics, which limit the underlying processes of any computing device made of matter. This approach reached the peak in the work of Fredkin and Toffoli on conservative logic published in 1982 [10].

The goal of this paper is to describe reversible circuits for the Karatsuba's algorithm and discuss their role in the general analysis performed by Bennett among others on reversible simulation of irreversible circuits [2, 3]. Existing analyses of reversible computing use a very general point of view mainly to establish generic results on the efficiency of reversible simulation of irreversible algorithms [3, 15, 16]. A general method to convert irreversible circuits into reversible ones is known as we discuss in the sequel, but it does not necessarily generate the best reversible simulation. An example of the inefficiency of the general method is provided by Fredkin and Toffoli (see pp. 232 and 233 of [10]). So, it is important to analyze each particular case to achieve the most efficient circuit. In general, the efficiency and the garbage disposal scheme are the main issues in the reversible simulation of irreversible circuits. We stress these issues throughout the paper.

The attempt to develop a garbage disposal scheme for Karatsuba's algorithm raises the following question: Is there a generic efficient algorithm to dispose of the garbage of recursive algorithms that follows their recursive structure? We point out that Bennett's schemes do not take into account the recursive structure of the original circuit.

A secondary motivation for developing reversible circuits for integer multiplication comes from their possible implementation in quantum computers. Quantum Computing has attracted a lot of attention in the last years, due to the possibility of building quantum devices for performing useful calculations that are exponentially faster than classical computers. The celebrated Shor's algorithm [19] is the best candidate for proving this exponential gap. Regarding integer multiplication, the algorithms developed so far for quantum computers are still based basically on the grade-school method [1, 20]. In an unpublished work, Zalka [23] discusses the Schönhage-Strassen's algorithm [18]. He also analyzes briefly the space usage of quantum Karatsuba's algorithms, and argues that it uses $O(n^{\log_2 3})$ of space, where n is the number of digits of the integers that are being multiplied. Based on assumptions, Zalka argues that Karatsuba's algorithm uses too much space to be useful in quantum computation. This is not what is usually expected since the crossover point (where the Schönhage-

Strassen's beats the Karatsuba's algorithm) is currently 2^{15} bits (approximately 10000 decimal digits). The crossover usually depends on the implementation details.[1]

The structure of this paper is as follows. In section 2, we outline the main results of the theory of reversible computation, stressing the garbage disposal schemes. In section 3, we discuss a garbage disposal scheme for recursive algorithms. In section 4, we review the classical version of Karatsuba's algorithm. In section 5, we present the reversible circuits for this algorithm. In the appendix we describe a detailed calculation of the steps of the reversible version of Karatsuba's algorithm for the multiplication of n-digit integers.

2 Reversible Computation

Landauer [13] showed that the removal of one bit of information at temperature $\mathbb{T}$ is performed by dissipating at least $k\mathbb{T}\ln 2$ in form of heat. He employed the well-known thermodynamical formulas $\delta S = \frac{\delta Q}{\mathbb{T}}$ and $S = k \ln W$, where S is the thermodynamical entropy, δQ is the heat introduced in the system by the environment which is kept at temperature $\mathbb{T}$, k is the Boltzmann constant, and W is the number of macrostates (equal to 2 for one bit).

On the other hand, reversible computation does not erase a bit of information, and so requires no expenditure of energy in principle. Fredkin and Toffoli [10] provided an insightful example of a reversible computer using billiard balls that are reflected by pieces of wall. The presence or the absence of a ball as input represents the bits 0 and 1, respectively. The balls are shot simultaneously (with some energy expenditure, which is recovered at the end). They are reflected by the walls properly placed and may collide with each other. The output is read by observing the final positions of the balls. Supposing that the collisions are perfect elastic, there is no dissipation of energy and the whole process is reversible. Fredkin and Toffoli showed how to build the Fredkin gate using their device. The Fredkin gate is universal, so the billiard ball computer can compute any partial recursive function reversibly, provided that an arbitrary large number of billiard balls and space are available. The catch is that collisions are not perfect elastic and there are extra interactions that cannot be totally disregarded. Errors would accumulate destroying the desired computation unless some correction code is used. Error correction codes destroy undesired information irreversibly.

The set of one and two-bit reversible gates is not universal [17]. On the other hand, the Toffoli gate is universal for reversible computation. Recall that the Toffoli gate has three bits, the first two are control bits, the outputs of which

---

[1] See http://magma.maths.usyd.edu.au/magma/Features/node86.html of Magma online help for crossover estimates and Zuras [24] for comparisons with other multiplication algorithms. Zuras states that Schönhage-Strassen's algorithm may never be best for any reasonably sized numbers.

are the same of the inputs, and third bit is the target that is flipped iff both control bits are set to 1. The usual proof for the universality of the Toffoli gate is indirect. The Toffoli gate simulates the irreversible gates FANOUT and NAND, if one disregards some outputs and keeps fixed some inputs. {FANOUT, NAND} is a universal set for irreversible computation; therefore the Toffoli gate alone is a universal set for reversible computation (ignoring some outputs and fixing some inputs). To build a reversible circuit, the procedure is the following: One builds beforehand an irreversible circuit, then replaces all occurrences of the FANOUT and NAND gates by the corresponding simulation in terms of the Toffoli gate. This is not a clean procedure because there are unwanted outputs that can be considered as some sort of information garbage. It is there simply to make the whole process reversible. The size of the garbage is $O(n)$, where n is the number of Toffoli gates, since for each Toffoli gate there are at most two bits of garbage. The garbage can be disposed of by erasing it, which produces at least $k\mathbb{T}\ln 2$ of heat per bit erased.

A generic method to dispose of the garbage with no expenditure of energy was initially proposed by Lecerf [14] and rediscovered by Bennett [2]. The diagram of figure 1 describes this method. Suppose one has built a reversible circuit to compute function f, which has the arguments INPUT, ZEROS, FIXED ANCILLA BITS and image INPUT, OUTPUT, GARBAGE. The desired output is copied using CNOT gates to some extra register. Recall that the CNOT gate has two bits, a control and a target. The target bit flips iff the control bit is set to 1. After copying the desired output with a CNOT for each bit, one can uncompute (or compute f^{-1}) by taking the arguments INPUT, OUTPUT, and GARBAGE. The output of the whole circuit has no garbage. The whole process is reversible and does not change the complexity of the original reversible circuit.

Bennett's schemes do not work for quantum computing algorithms, due to

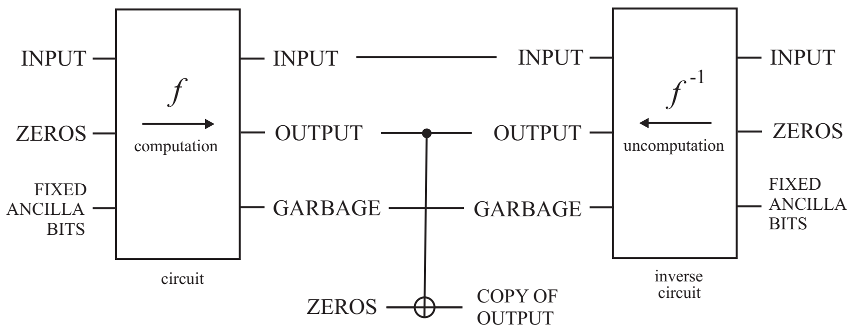

Figure 1: Outline of Bennett's scheme to dispose of the garbage with no energy expenditure. The "inverse circuit" is obtained by inverting the first "circuit".

the no-cloning theorem [22]. The CNOTs do not produce a copy of OUTPUT when it is in a superposition state.

Bennett's first scheme (figure 1) runs in time $\Theta(T)$ and space $\Theta(T+S)$, where T and S are the time and space bounds of the irreversible circuit which is being simulated reversibly. The space bound is troublesome when one simulates an irreversible circuit which is exponential in time but polynomial in space. The simulation will be exponential both in time and space. This problem was addressed again by Bennett [3], who improved his earlier scheme, proposing a new generic scheme which runs in time $O(T^{1+\epsilon})$ and space $O(T \log S)$ given any $\epsilon > 0$. Levine and Sherman[15] showed that there is a hidden factor in the space bound of the form $\epsilon 2^{1/\epsilon}$, which diverges as ϵ approaches 0. Using recurrence equations, they proved that Bennett's second scheme actually runs in time $\Theta(T^{1+\epsilon}/S^{\epsilon})$ and space $\Theta(S(1+\log \frac{T}{S}))$.

Bennett's second scheme is ingenious. The irreversible circuit is split into m blocks of length S approximately so that $mS \approx T$. The goal is to use less space than the first scheme. If one runs n consecutive blocks, the space usage is nS. The idea is to run these steps backward disposing of the garbage and reusing the same space for the next blocks. The major problem is that to run the i-th block one needs to know the output of $(i-1)$-th block, and vice-versa, to run the $(i-1)$-th block backwards, one needs the output of the i-th block. Some checkpoint blocks must be left to proceed with the reversible simulation. After running another set of blocks, one wants to remove the previous checkpoint, which requires to rerun the simulation from the beginning up to that checkpoint. This leads to a hierarchical recursive scheme. The details can be found in [3, 16].

3 Garbage Disposal in Recursive Algorithms

Bennett's second scheme does not consider any recursive structure that the original algorithm might have. In the scheme, the algorithm is split in roughly equal segments of size S (the space bound of the original algorithm) and the segments are run forward and backward in a hierarchical manner having no relation to any recursive structure of the original algorithm. We pose the following question: Can one simulate reversibly and efficiently an irreversible recursive circuit such that the garbage disposal follows the recursive structure of the original algorithm?

Now we discuss a scheme to dispose of the garbage at each level of the recursion. Such scheme is much simpler to employ for recursive algorithms than Bennett's second scheme. Unfortunately, a general analysis shows that this recursive scheme may have greater space complexity. To analyze the complexity, we employ the Master method [8]. If the recursive algorithm divides a problem of size n into a instances, each of size $\frac{n}{b}$, where a and b are positive constants, then the running time is

$$T(n) = aT\left(\frac{n}{b}\right) + f(n),$$

where $f(n)$ is a positive function that can be explicitly bounded. According to the Master theorem, there are three cases depending on the asymptotic behavior of $f(n)$:

1. $T(n) = \Theta(n^{\log_b a})$, if $f(n) = O(n^{\log_b a-\epsilon})$ for some $\epsilon > 0$.
2. $T(n) = \Theta(n^{\log_b a} \log n)$, if $f(n) = \Theta(n^{\log_b a})$.
3. $T(n) = \Theta(f(n))$, if $f(n) = \Omega(n^{\log_b a-\epsilon})$ for some $\epsilon > 0$, and if $af(\frac{n}{b}) \leq cf(n)$ for some $c < 1$ and all sufficiently large n.

The recursive garbage removal scheme is as follows. One employs Bennett's first scheme in each level of the recursion. One simply describes the garbage removal when the input is n. It is recursively used in the recursive calls. A concrete example is provided in figure 4 for the Karatsuba's algorithm. The drawback of this scheme is that it doubles the number of gates per recursive level, therefore $T(\frac{n}{b})$ goes to $2T(\frac{n}{b})$. Using the Master theorem and replacing a by $2a$, we conclude that the complexity of the simulation would be $\Theta(T(n)\ n^{\log_b a})$ for the cases 1 and 2. In case 3, the complexity would not change, since it is determined by the behavior of $f(n)$. Usually $b \geq 2$; therefore for important classes of recursive algorithms, the running time of this simulation would be polynomially greater than the corresponding one using Bennett's usual scheme.

4 Karatsuba's Algorithm

Karatsuba [12] discovered the first algorithm that accomplishes the multiplication of multiprecision integers of size n digits with complexity below that of the usual $O(n^2)$. The complexity of the Karatsuba's algorithm is $O(n^{\log_2 3})$, which can be proved easily by using a recurrence relation, as described below.

Consider two n-bit integers f and g. Suppose that $n = 2^m$ for some positive integer m. We can split f into two equal parts f_1 and f_0, and g into g_1 and g_0 such that

$$f = f_1 2^{\frac{n}{2}} + f_0,$$
$$g = g_1 2^{\frac{n}{2}} + g_0.$$

Karatsuba's method uses the following identity

$$fg = f_1 g_1 2^n + ((f_1 + f_0)(g_1 + g_0) - f_1 g_1 - f_0 g_0) 2^{\frac{n}{2}} + f_0 g_0. \tag{1}$$

There are more additions in this identity than in the usual multiplication, but the number of recurrent multiplications is lesser. If $T(n)$ is the number of operations, then the cost to multiply two n-digit integers using equation (1) is

$$T(n) = 3T\left(\frac{n}{2}\right) + \alpha n,$$

where α is some constant. The solution of this recurrence equation is $T(n) = \alpha' n^{\log_2 3}$. The constant α' is larger than the corresponding one in the usual quadratic multiplication algorithm (see [24] for estimates of the value of α'). This implies that the crossover between these algorithms occurs for integers with large number of digits (between 2^7 and 2^8 bits depending on the implementation details). Algorithms for integer multiplication used nowadays in computer algebra systems such as Maple are based on Karatsuba's algorithm. See references [4, 6, 21] for reviews and recent improvements on this algorithm.

5 Reversible Karatsuba's Algorithm

Figure 2 describes an intermediate circuit called KARATSUBA. It is a recursive reversible version of the Karatsuba's algorithm. The gates KARATSUBA $\lceil \frac{n}{2} \rceil$ and KARATSUBA $\lceil \frac{n}{2} \rceil + 1$ must be replaced by a circuit similar to the original one, but the input size is half or half plus one of the original. As soon as the number of digits in the recursive call is less than the crossover point, the usual reversible multiplication algorithm is issued, halting the recursive loop. In this case we use some multiplication gate already known, such as those described in [1, 20]. We also employ the reversible addition gate as described in [1, 9, 20]. In figure 2 this gate is called ADD n, where n is the number of bits of the integers that are being added. As usual, the black stripe shows the direction of the gate. The usual direction computes the addition, while the opposite direction computes the subtraction if the first operand is greater than the second (first means the rightmost line). One can check in figure 2 that the first operand of the inverted ADD is $(f_1 + f_0)(g_1 + g_0)$ and the second is $f_1 g_1 + f_0 g_0$. Eq. (1) tells us that $(f_1 + f_0)(g_1 + g_0) \geq f_1 g_1 + f_0 g_0$, therefore the inverted ADD performs the usual subtraction. We call this version parallel because there are 3 recursive calls that can be performed simultaneously.

Note the presence of garbage bits in KARATSUBA. To dispose of this garbage, we employ Bennett's first scheme, copying the result to an extra register and running the algorithm KARATSUBA backward. Figure 3 shows the circuit of this algorithm, which we call KARATSUBA(1). Its complexity is described in the upper left box of table 1. Note that there are three parallel recursive calls of the KARATSUBA gate. This is a strategy to decrease the time complexity to $O(n)$, paying the price of increasing the space complexity to $O(n^{\log_2 3})$. This trade-off seems optimal if one compares to the complexity of the usual implementation of the irreversible version, which is $T(n) = O(n^{\log_2 3})$ and $S(n) = O(n)$. This strategy guarantees that Bennett's first scheme is worthwhile in this case.

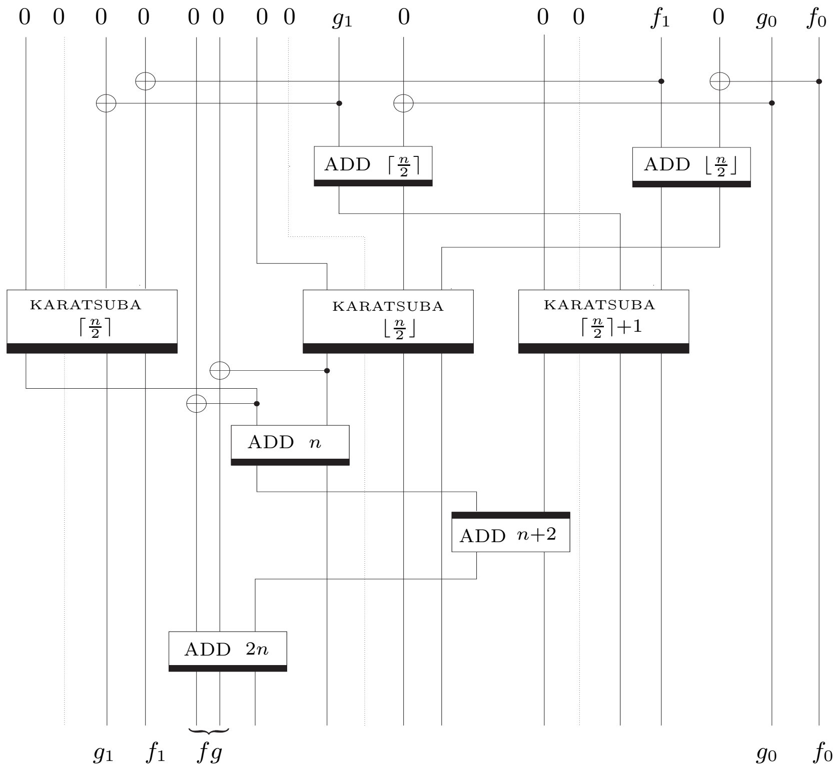

Figure 2: The KARATSUBA circuit. It is a parallel version of the reversible circuit for the Karatsuba's multiplication algorithm. The circuit is defined recursively. Note the KARATSUBA gates running in parallel with the inputs strictly smaller than the original inputs, which are the n-bit integers $g = g_1 g_0$ and $f = f_1 f_0$. There is garbage in the output. The lowermost ADD gate has 3 inputs because the 2 inputs on the left join to form the second operand. These 2 inputs are the first and third terms in the right hand side of equation (1). Dotted lines refer to the work bits in the recursive calls.

Figure 4 shows an example of recursive garbage disposal scheme discussed in section 3. This circuit uses a slight variation of the Bennett's first scheme at each recursive level. Note that figures 2 and 4 are different in the following points: in figure 2, the multiplication is performed up to the end with no garbage removal and in figure 4, the garbage is removed while the multiplication is being performed. The complexity of KARATSUBA(2) described in figure 4 is given in the lower left box of table 1. As discussed in section 3, this garbage removal scheme is very elegant but unfortunately it almost doubles the number of gates in each recursive level, increasing significantly the complexity of the reversible simulation. Note that KARATSUBA(2) also uses parallel recursive calls. This means that the three recursive calls can be run simultaneously decreasing the complexity in time. Being rigorous, the garbage removal scheme is a slight varia-

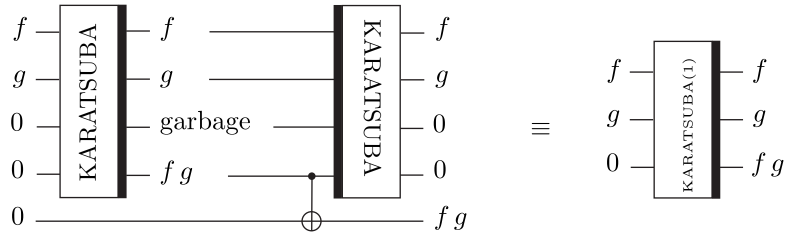

Figure 3: The KARATSUBA(1) circuit. It is the garbage-free version of the reversible recursive KARATSUBA circuit described in figure 2. It uses Bennett's first scheme to dispose of the garbage. Its complexity is given in table 1. The output 0 is not shown in KARATSUBA(1).

garbage disposal scheme	position of multiplication gates	
	parallel	sequential
Bennett's first scheme	KARATSUBA(1) $T(n) = O(n)$ $S(n) = O(n^{\log_2 3})$ $N(n) = O(n^{\log_2 3})$	KARATSUBA(3) $T(n) = O(n^{\log_2 3})$ $S(n) = O(n^{\log_2 3})$ $N(n) = O(n^{\log_2 3})$
recursive scheme	KARATSUBA(2) $T(n) = O(n \log n)$ $S(n) = O(n^{\log_2 3})$ $N(n) = O(n^{\log_2 6})$	KARATSUBA(4) $T(n) = O(n^{\log_2 6})$ $S(n) = O(n)$ $N(n) = O(n^{\log_2 6})$

Table 1: Complexity of the reversible simulations of the Karatsuba's algorithm. T and S are time and space bounds respectively. N is the number of universal gates (Toffoli gates).

tion of Bennett's first scheme, because the gate ADD $2n$ in the center of figure 4 was not duplicated. In this case we need not to use CNOT gates to make a copy of the output.

Table 1 shows the complexity of two other versions that we have analyzed but have not displayed the circuits. These versions (KARATSUBA(3) and KARATSUBA(4)) can be obtained respectively from figures 2 and 4 by changing the positions of the recursive calls in such way that they are run in sequence. In this case, the circuit uses less space, $S(n) = O(n^{\log 3})$ for KARATSUBA(3) and $S(n) = O(n)$ for KARATSUBA(4). They use less space because the work bits used by first gate KARATSUBA $\lceil \frac{n}{2} \rceil$ are reused by the second KARATSUBA $\lceil \frac{n}{2} \rceil$, and reused again by KARATSUBA $\lceil \frac{n}{2} \rceil + 1$.

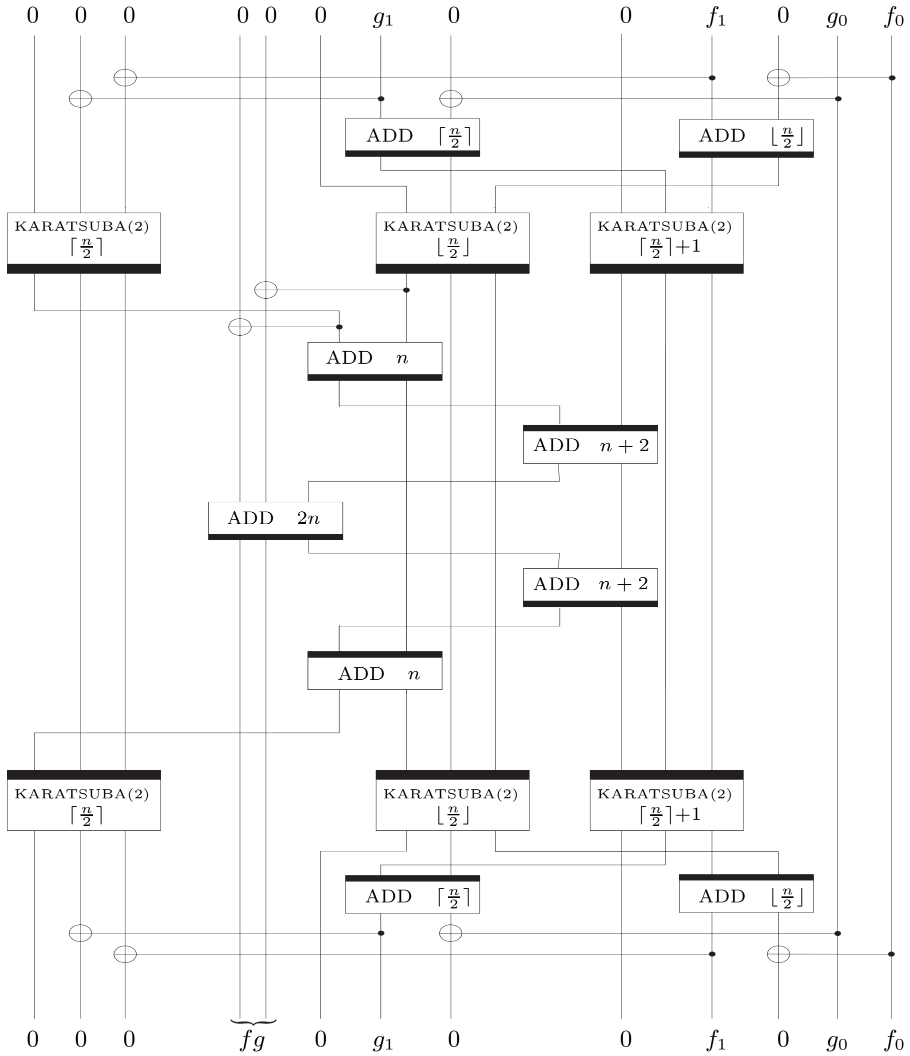

Figure 4: The KARATSUBA(2) circuit. It uses a recursive garbage disposal scheme. The gate KARATSUBA(2) $\lceil\frac{n}{2}\rceil$ must be replaced by a circuit similar to the one described in the figure, but the input size is half of the original. Note that this replacement has almost twice more gates than the one which is performed in figure 2.

6 Conclusions

We have analyzed the computational complexity of four recursive versions of the Karatsuba's algorithm. KARATSUBA(4) is the worst in complexity, worse than the grade-school method. It is the best candidate to employ Bennett's second scheme instead of the recursive scheme. KARATSUBA(2) and KARATSUBA(3) use the same space but KARATSUBA(2) is a little better than KARATSUBA(3) in time. KARATSUBA(1) is the best one even using Bennett's first scheme (instead of the second) because the complexity in time is $O(n)$. The overhead introduced by the first scheme is not relevant compared to the complexity in space, which is $S(n) = O(n^{\log 3})$.

An important issue in reversible simulations is the garbage removal scheme, as Bennett pointed out. We have not used Bennett's second scheme because it does not match the recursive structure of the Karatsuba's algorithm. It would be cumbersome to describe the circuit details in this case, because some lower level circuitry would be required. Besides, the overall complexity would not be better than the complexity of KARATSUBA(1).

Acknowledgements

We thank Valmir Barbosa, Raul Donangelo, and Carlile Lavor for useful discussions. We also thank the financial support of CNPq.

References

1. D. Bechman, A. N. Chari, S. Devabhaktuni, and J. Preskill. Efficient networks for quantum factoring. *Phys. Rev. A*, 54:1034–1063, 1996.
2. C. H. Bennett. The logical reversibility of computation. *IBM J. Res. Develop.*, 17:525–532, 1973.
3. C. H. Bennett. Time/space trade-offs for reversible computation. *SIAM J. Comput.*, 18(4):766–776, 1989.
4. D.J. Bernstein. Fast multiplication and its applications. http://cr.yp.to/papers.html, 2003.
5. G. Cesari and R. Maeder. Performance analysis of the parallel Karatsuba multiplication algorithm for distributed memory architectures. *J. Symb. Comput.*, 21(4-6):467–473, 1996.
6. N.S. Chang, C.H. Kim, Y.-H. Park, and J. Lim. A non-redundant and efficient architecture for Karatsuba-Ofman algorithm. In *ISC*, pages 288–299, 2005.
7. L.S. Cheng, A. Miri, and T.H. Yeap. Improved FPGA implementations of parallel Karatsuba multiplication over GF($2n$). In *In 23rd Biennial Symposium on Communications*, 2006.
8. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms.* MIT Press, 2nd edition, 2001.
9. T. G. Draper. Addition on a quantum computer. www.arxiv.org, 2000. quant-ph/0008033.
10. E. Fredkin and T. Toffoli. Conservative Logic. *Internat. J. Theoret. Phys.*, 21:219–253, 1982.
11. T. Jebelean. Using the parallel Karatsuba algorithm for long integer multiplication and division. In *European Conference on Parallel Processing*, pages 1169–1172, 1997.
12. A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics – Doklady*, 7(7):595–596, 1963.
13. R. Landauer. Irreversibility and heat generation in the computing process. *IBM J. Res. Develop.*, 5:183–191, 1961.
14. M.Y. Lecerf. Machines de Turing réversibles. Récursive insolubilité en $n \in N$ de l'équation $u = \theta^n u$, où θ est un "isomorphisme de codes". *Comptes Rendus*, 257:2597–2600, 1963.

15. R.Y. Levine and A. T. Sherman. A note on Bennett's time-space tradeoff for reversible computation. *SIAM J. Comput.*, 19(4):673–677, 1990.
16. M. Li, J. Tromp, and P. Vitnyi. Reversible simulation of irreversible computation. *Physica D*, 120:168–176, 1998.
17. J. Preskill. *Lecture Notes for Physics 229: Quantum Information and Computation.* California Institute of Technology, September 1998.
18. A. Schönhage and V. Strassen. Schnelle multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.
19. P. Shor. Algorithms of quantum computation: Discrete logarithm and factoring. In *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.
20. V. Vedral, A. Barenco, and A. Ekert. Quantum networks for elementary arithmetic operations. *Physical Review A*, 54(1):147–153, 1996.
21. A. Weimerskirch and C. Paar. Generalizations of the Karatsuba algorithm for efficient implementations. Technical report, 2003.
22. W. K. Wooters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, 1982.
23. C. Zalka. Fast version of Shor's quantum factoring algorithm. 1998. quant-ph/9806084.
24. D. Zuras. More on squaring and multiplying large integers. *IEEE Transactions on computers*, 43(8):899–908, 1994.

Appendix A

IN	0	0	0	0	0	0	g_1	0	0	f_1	0	g_0	f_0
1	0	g_1	f_1	0	0	0	g_1	g_0	0	f_1	f_0	g_0	f_0
2	0	g_1	f_1	0	0	0	g_1+g_0	g_0	0	f_1+f_0	f_0	g_0	f_0
3	g_1f_1	g_1	f_1	0	0	0	g_1+g_0	g_0	$(g_1+g_0)\cdot(f_1+f_0)$	f_1+f_0	f_0	g_0	f_0
4	g_1f_1	g_1	f_1	g_1f_1	g_0f_0	0	g_1+g_0	g_0	$(g_1+g_0)\cdot(f_1+f_0)$	f_1+f_0	f_0	g_0	f_0
5	$g_1f_1+g_0f_0$	g_1	f_1	g_1f_1	g_0f_0	0	g_1+g_0	g_0	$(g_1+g_0)\cdot(f_1+f_0)$	f_1+f_0	f_0	g_0	f_0
6	$(g_1+g_0)(f_1+f_0)$ $-(g_1f_1+g_0f_0)$	g_1	f_1	g_1f_1	g_0f_0	0	g_1+g_0	g_0	$(g_1+g_0)\cdot(f_1+f_0)$	f_1+f_0	f_0	g_0	f_0
7	$(g_1+g_0)(f_1+f_0)$ $-(g_1f_1+g_0f_0)$	g_1	f_1	$f_1g_12^n+((f_1+f_0)(g_1+g_0)-f_1g_1-f_0g_0)2^{\frac{n}{2}}+f_0g_0$		0	g_1+g_0	g_0	$(g_1+g_0)\cdot(f_1+f_0)$	f_1+f_0	f_0	g_0	f_0
OUT	garbage	g_1	f_1	fg		garbage						g_0	f_0

Table 2: Detailed evolution of all registers of the KARATSUBA circuit outlined in figure 2, with the exception of the ones represented by dotted lines.

Table 2 describes the detailed evolution of all registers of the KARATSUBA circuit outlined in figure 2. There are 7 steps. Gates that can be applied simultaneously are counted as one step. In table 2, we have not interchanged the position of the registers. The input are the integers f and g which are split into f_0, f_1 and g_0, g_1 respectively. We are supposing that the number of digits of f_1 and g_1 is $\lceil \frac{n}{2} \rceil$ and the number of digits of f_0 and g_0 is $\lfloor \frac{n}{2} \rfloor$. Care must be taken in order to read the input of the lowermost ADD gate. This gate performs the addition of the terms of equation (1). The output fg is displayed at the bottom of the fifth column of table 2.