VLSI DESIGN OF ITERATIVE KARATSUBA MULTIPLIER AND ITS EVALUATION

Syunji Yazaki and Kôki Abe
Department of Computer Science, The University of Electro-Communications
1-5-1 Chofugaoka Chofu-shi,Tokyo 182-8585 Japan
{syunji, abe}@cacao.cs.uec.ac.jp

ABSTRACT

Multi-digit multiplication is widely used for various applications in recent years, including numerical calculation, chaos arithmetic, primality testing. Systems with high performance and low energy consumption are demanded, especially for image processing and communications with cryptography using chaos. In this paper, hardware design of multi-digit integer multiplication is described and its VLSI realization is evaluated in terms of the cost and performance. A version of Karatsuba hardware using 0.18μm process can perform 512-bit multiplications 30 times faster than software at the area cost of 6.91mm^2. Computation energy was found to be nearly 10^{-3} of that consumed by general purpose processor which executes the software version . The results obtained by this study will help system designers for applications requiring multi-digit multiplication to select design alternatives including ASIC realization.

KEY WORDS

multi-digit multiplication, Karatsuba, VLSI, performance, area cost, energy consumption

1 Introduction

Many studies have been done for enhancing system performance by employing dedicated hardware for computations frequently used in specific applications. Reducing the power consumption is another motivation for dedicated hardware. Using a small scale ASIC (Application Specific Integrated Circuit) with less power instead of using general purpose processor is justified in many modern applications such as secured mobile communications.

Multi-digit multiplication is widely used for various applications in recent years, including numerical calculation[1], chaos arithmetic[2], primality testing[3]. In particular, cryptography using chaos found many applications in image processing[4] and communications[5]. For such applications, systems with high performance and low energy consumption are demanded.

Two well-known methods of realizing high-performance multi-digit multiplication exist: Karatsuba method with computation complexity of $O(n^{1.58})$[6] and FFT method with the complexity of $O(n \log n \log \log n)$[7], where n stands for bit length of operands. Karatsuba method is employed in multiplication of hundreds to thousands bits, whereas FFT method is used for millions bit multiplication. We have reported the results of hardware implementation of FFT method in Ref.[8].

There are many studies on hardware implementation of Karatsuba algorithm over Galois Field (GF) intended for use in elliptic curve cryptography [9, 10]. Regarding hardware implementation of multi-digit integer multiplication based on Karatsuba algorithm for applications such as chaotic cryptography and communications, however, studies are rare. Among them Ref.[11] dealt with hardware implementation of 32-bit integer Karatsuba multiplier, and Ref.[12] designed and evaluated integer Karatsuba multipliers of up to 512-bit length with combinational circuits.

In this paper we investigate the cost and performance of hardware implementation of multi-digit integer multiplication of ranging from 16 to more than 512 bits with sequential circuit. (We use the term "multi-digit multiplication" for multi-digit integer multiplication hereafter.) Our contributions are mainly to allow system designers to select design alternatives including ASIC realization for specific applications requiring multi-digit multiplication.

The rest of this paper is organized as follows. Section 2 briefly explains Karatsuba algorithm. Section 3 present design and implementation of hardware Karatsuba multiplier based on iterative approach. In Section 4, evaluation results of the design implemented on an ASIC are described with discussions. Section 5 closes with a summary.

2 Karatsuba Algorithm

Let A and B be $2n$-bit integers. We split them into halves as $A = a_1 2^n + a_0$ and $B = b_1 2^n + b_0$. Product P of A and B is given by

$$\begin{aligned} P = A \cdot B &= (a_1 2^n + a_0) \cdot (b_1 2^n + b_0) \\ &= a_1 \cdot b_1 2^{2n} + (a_1 \cdot b_0 + a_0 \cdot b_1) 2^n + a_0 \cdot b_0. \end{aligned} \quad (1)$$

Using the relation $a_1 \cdot b_0 + a_0 \cdot b_1 = (a_1 + a_0) \cdot (b_1 + b_0) - (a_1 \cdot b_1 + a_0 \cdot b_0)$, we can rewrite eq.(1) as

$$\begin{aligned} P = &a_1 \cdot b_1 2^{2n} \\ &+ ((a_1 + a_0) \cdot (b_1 + b_0) - (a_1 \cdot b_1 + a_0 \cdot b_0)) 2^n \\ &+ a_0 \cdot b_0. \end{aligned} \quad (2)$$

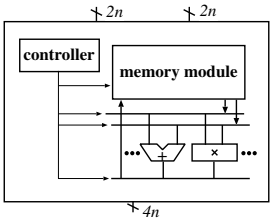

Figure 1. Iterative Karatsuba multiplier (IKM).

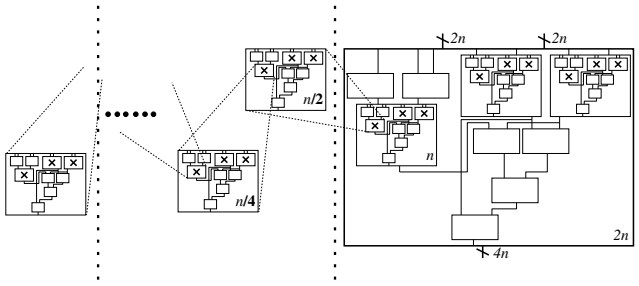

Figure 2. Recursive Karatsuba multiplier (RKM).

Number of n-bit multiplications required for multiplying $2n$-bit integers is three instead of four, when we use eq.(2) instead of eq.(1). Karatsuba algorithm uses eq.(2) for multiplication recursively, resulting in the complexity of $O(n^{1.58})$, where $\log_2 3 \approx 1.58$. In software implementation, the algorithm is applied recursively up to multiplications of a word length.

When implementing Karatsuba algorithm by hardware, there are two alternatives: iterative and recursive approaches. In the iterative approach the algorithm is realized by sequential circuits as shown in Fig.1. We call this architecture as iterative Karatsuba multiplier (IKM). In the recursive approach the algorithm is realized by combinational circuits as shown in Fig.2. We call this architecture as recursive Karatsuba multiplier (RKM). The complexity $O(n^{1.58})$ of Karatsuba method is converted to time complexity in IKM and to area complexity in RKM. Multiplier for integers of longer than thousands bits must be realized by IKM because the circuit size would become huge if RKM were used. In this paper we deal with the VLSI design of IKM and evaluate its area cost and performance.

3 Design of Iterative Karatsuba Multiplier

3.1 Approach

Here we describe an approach for implementing Karatsuba algorithm with sequential circuits. The approach is based on Ref.[10] where hardware Karatsuba multiplication on GF was studied. In the following we consider the case of applying Karatsuba algorithm two times recursively to $4n$-bit multiplications by using n-bit multipliers as fundamental combinational circuits, where n is referred to as basic bit length.

Multiplier A and multiplicand B each of $4n$-bit length are divided into four n-bit segments each

$$A = a_3 2^{3n} + a_2 2^{2n} + a_1 2^n + a_0, \tag{3}$$
$$B = b_3 2^{3n} + b_2 2^{2n} + b_1 2^n + b_0. \tag{4}$$

Let P be the products of A and B:

$$P = p_7 2^{7n} + p_6 2^{6n} + p_5 2^{5n} + p_4 2^{4n} + p_3 2^{3n} + p_2 2^{2n} + p_1 2^n + p0 \tag{5}$$

Let $A_1 = a_3 2^n + a_2$, $A_0 = a_1 2^n + a_0$、 $B_1 = b_3 2^n + b_2$, $B_0 = b_1 2^n + b_0$. Then $A = A_1 2^{2n} + A_0$, $B = B_1 2^{2n} + B_0$, and the product P is given by

$$P = AB = A_1 B_1 2^{4n} + \big(A_{10} B_{10} - A_1 B_1 - A_0 B_0\big) 2^{2n} + A_0 B_0, \tag{6}$$

where $A_{10} = A_1 + A_0$ and $B_{10} = B_1 + B_0$. In what follows, a sum of two variables is denoted by a symbol with suffices concatenating those of the variables to be added as above. Each term in the expression eq.(6), $A_1 B_1$, $A_0 B_0$, $A_{10} B_{10}$, is expanded as follows:

$$\begin{aligned} A_1 B_1 &= (a_3 2^n + a_2)(b_3 2^n + b_2), \\ &= a_3 b_3 2^{2n} + (a_{32} b_{32} - a_3 b_3 - a_2 b_2) 2^n + a_2 b_2, \end{aligned} \tag{7}$$

$$\begin{aligned} A_0 B_0 &= (a_1 2^n + a_0)(b_1 2^n + b_0), \\ &= a_1 b_1 2^{2n} + (a_{10} b_{10} - a_1 b_1 - a_0 b_0) 2^n + a_0 b_0, \end{aligned} \tag{8}$$

$$\begin{aligned} A_{10} B_{10} &= (a_{31} 2^n + a_{20})(b_{31} 2^n + b_{20}), \\ &= (a_{31})(b_{31}) 2^{2n}, \\ &+ (a_{3120} b_{3120} - a_{31} b_{31} - a_{20} b_{20}) 2^n, \\ &+ a_{20} b_{20}. \end{aligned} \tag{9}$$

Substituting eqs.(7)(8)(9) into eq.(6), we get

$$\begin{aligned} P = {} & 2^{6n} a_3 b_3 \\ & + 2^{5n} \big(a_{32} b_{32} - a_3 b_3 - a_2 b_2\big) \\ & + 2^{4n} \big(a_{31} b_{31} - a_3 b_3 - a_1 b_1 + a_2 b_2\big) \\ & + 2^{3n} \big(a_{3120} b_{3120} - a_{31} b_{31} - a_{20} b_{20} - a_{32} b_{32} + a_3 b_3 + a_2 b_2\big) \\ & + 2^{2n} \big(a_{20} b_{20} - a_2 b_2 - a_0 b_0 + a_1 b_1\big) \\ & + 2^n \big(a_{10} b_{10} - a_1 b_1 - a_0 b_0\big) \\ & + a_0 b_0. \end{aligned} \tag{10}$$

Relations between coefficients $p7,\ldots,p0$ in eq.(5) and partial products $a_0 b_0$, ..., $a_3 b_3$, $a_{10} b_{10}$, ..., $a_{32} b_{32}$, $a_{3120} b_{3120}$ in expressions of eq.(10) are summarized in Table 1. Rows L and H denote lower n-bit part and the remaining higher part of partial products, respectively. The coefficients $p7,\ldots,p0$ of the product P are obtained by accumulating the partial products according to the table. Accumulations are carried out so that a carry produced in an accumulation for a coefficient be added to that for the adjacent higher order coefficient. The case of applying Karatsuba algorithm more than two times recursively can

Table 1. Relations between coefficients in eq.(5) and partial products. Rows L and H denote lower n-bit part and the remaining higher part of partial products, respectively.

Partial Products			p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0
$pp0$	a_0b_0	L					$+$	$-$	$-$	$+$
		H				$+$	$-$	$-$	$+$	
$pp1$	a_1b_1	L				$-$	$+$	$+$	$-$	
		H			$-$	$+$	$+$	$-$		
$pp2$	a_2b_2	L			$-$	$+$	$+$	$-$		
		H		$-$	$+$	$+$	$-$			
$pp3$	a_3b_3	L		$+$	$-$	$-$	$+$			
		H	$+$	$-$	$-$	$+$				
$pp4$	$a_{10}b_{10}$	L					$-$		$+$	
		H				$-$		$+$		
$pp5$	$a_{20}b_{20}$	L					$-$	$+$		
		H				$-$	$+$			
$pp6$	$a_{31}b_{31}$	L				$+$	$-$			
		H			$+$	$-$				
$pp7$	$a_{32}b_{32}$	L			$+$		$-$			
		H		$+$		$-$				
$pp8$	$a_{3120}b_{3120}$	L					$+$			
		H				$+$				

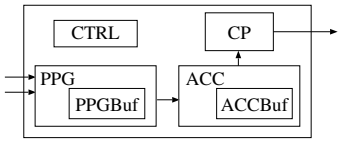

Figure 3. Structure of R2IKM, an IKM with two time recursions.

be described similarly. For instance, in the case of applying Karatsuba three times recursively. the table of 27 rows by 16 columns is required for calculating coefficients from partial products.

3.2 Structure of IKM and Its Modules

We design an IKM with two time recursions as an example. The design is denoted by R2IKM referring to the number of recursion times, and is based on the approach described above.

The structure of R2IKM is shown in Fig.3. The design consists of three modules, PPG (Partial Product Generator), ACC (Accumulator) and CP (Carry Propagator), and a control module CTRL (Controller). Details of the modules are described in the following. Basically the IKM of more than two time recursions has the same structure as the one shown in Fig.3.

3.2.1 Partial Product Generator Module (PPG)

The module PPG consisting of adders, a multiplier and buffer generates the partial products as shown in Fig.4. For the basic combinational multiplier used in PPG we employ WTM (Wallace Tree Multiplier) according to the results of our study[12]. As shown in the figure we use only one multiplier for PPG. This is because we intend cost-

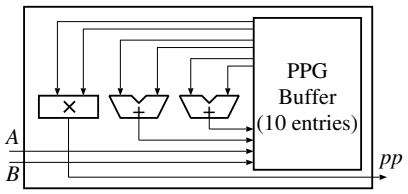

Figure 4. Structure of PPG which generates partial products.

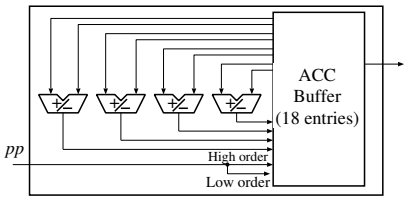

Figure 5. Structure of ACC which accumulates partial products.

preferred design. There is another design choice of using more than two WTMs for performance-preferred design, where partial products will be generated faster. The number of pipeline stages of WTM is adjusted to be six to balance its critical path with that of adders. We let a synthesis tool generate WTM using Design Ware Library provided by Synopsys Inc. The library generates WTM with booth recode by default except for multipliers of small bit length.

Two adders are used in the PPG module. This is because the area cost of adders is small compared to that of multipliers and buffers. Another reason is that two adders are required to operate the multiplier efficiently. Adders are generated by the synthesizer using Design Ware Library as well.

For buffering intermediate results of partial products, a buffer (PPG Buffer) is provided with I/O ports dedicated to each of the multiplier and adders. Writing into the buffer is performed synchronously, while reading from the buffer is performed asynchronously to operate the pipeline of PPG Buffer efficiently. The capacity of the buffer is estimated to be 10 entry as its minimum. Inputs A and B to PPG are also buffered to PPG.

Note that the bit length of values dealt with inside the module PPG is a few bits lager than that of inputs due to carries generated by addition.

3.2.2 Accumulator Module (ACC)

The module ACC consisting of adder-subtracters and a buffer accumulates partial products to generate coefficients $p0 \ldots p7$. The structure of ACC is shown in Fig.5.

Four adder-subtracters are used, taking into account the number of operations required to be performed for one partial product referring to Table 1. Another design choice is to increase the number of adder-subtracters to enhance the performance at the area cost of complex control. The adder-subtracters are synthesized using the Design Ware

Table 2. Scheduling of operations performed by PPG.

		Multiplier1		Adder1		Adder2	
Step	Input	Input	Output	Input	Output	Input	Output
1	a_0, b_0	–	–	–	–	–	–
2	a_1, b_1	a_0, b_0	–	–	–	–	–
3	a_2, b_2	a_1, b_1	–	a_1, a_0	–	b_1, b_0	–
4	a_3, b_3	a_2, b_2	–	a_2, a_0	a_{10}	b_2, b_0	b_{10}
5	–	a_3, b_3	–	a_3, a_1	a_{20}	b_3, b_1	b_{20}
6	–	a_{10}, b_{10}	–	a_3, a_2	a_{31}	b_3, b_2	b_{31}
7	–	a_{20}, b_{20}	–	a_{20}, a_{31}	a_{32}	b_{20}, b_{31}	b_{32}
8	–	a_{31}, b_{31}	$a_0 b_0 (= pp0)$	–	a_{2031}	–	b_{2031}
9	–	a_{32}, b_{32}	$a_1 b_1 (= pp1)$	–	–	–	–
10	–	a_{2031}, b_{2031}	$a_2 b_2 (= pp2)$	–	–	–	–
11	–	–	$a_3 b_3 (= pp3)$	–	–	–	–
12	–	–	$a_{10} b_{10} (= pp4)$	–	–	–	–
13	–	–	$a_{20} b_{20} (= pp5)$	–	–	–	–
14	–	–	$a_{31} b_{31} (= pp6)$	–	–	–	–
15	–	–	$a_{32} b_{32} (= pp7)$	–	–	–	–
16	–	–	$a_{2031} b_{2031} (= pp8)$	–	–	–	–

Table 3. Scheduling of operations performed by ACC.

		Adder-Subtracter1		Adder-Subtracter2		Adder-Subtracter3		Adder-Subtracter4	
Step	Input	Input	Output	Input	Output	Input	Output	Input	Output
1	$pp0$	–	–	–	–	–	–	–	–
2	$pp1$	$p0, pp0\mathrm{L}$	–	$p1, pp0\mathrm{L}$	–	$p2, pp0\mathrm{L}$	–	$p3, pp0\mathrm{L}$	–
3	$pp2$	$p1, pp0\mathrm{H}$	$p0 + pp0\mathrm{L}$	$p2, pp0\mathrm{H}$	$p1 - pp0\mathrm{L}$	$p3, pp0\mathrm{H}$	$p2 - pp0\mathrm{L}$	$p4, pp0\mathrm{H}$	$p3 + pp0\mathrm{L}$
4	$pp3$	$p1, pp1\mathrm{L}$	$p1 + pp0\mathrm{H}$	$p2, pp1\mathrm{L}$	$p2 - pp0\mathrm{H}$	$p3, pp1\mathrm{L}$	$p3 - pp0\mathrm{H}$	$p4, pp1\mathrm{L}$	$p4 + pp0\mathrm{H}$
5	$pp4$	$p2, pp1\mathrm{H}$	$p1 - pp1\mathrm{L}$	$p3, pp1\mathrm{H}$	$p2 + pp1\mathrm{L}$	$p4, pp1\mathrm{H}$	$p3 + pp1\mathrm{L}$	$p5, pp1\mathrm{H}$	$p4 - pp1\mathrm{L}$
6	$pp5$	$p2, pp2\mathrm{L}$	$p2 - pp1\mathrm{H}$	$p3, pp2\mathrm{L}$	$p3 + pp1\mathrm{H}$	$p4, pp2\mathrm{L}$	$p4 + pp1\mathrm{H}$	$p5, pp2\mathrm{L}$	$p5 - pp1\mathrm{H}$
7	$pp6$	$p3, pp2\mathrm{H}$	$p2 - pp2\mathrm{L}$	$p4, pp2\mathrm{H}$	$p3 + pp2\mathrm{L}$	$p5, pp2\mathrm{H}$	$p3 + pp2\mathrm{L}$	$p6, pp2\mathrm{H}$	$p4 - pp2\mathrm{H}$
8	$pp7$	$p3, pp3\mathrm{L}$	$p3 - pp2\mathrm{H}$	$p4, pp3\mathrm{L}$	$p4 + pp2\mathrm{H}$	$p5, pp3\mathrm{L}$	$p5 + pp2\mathrm{H}$	$p6, pp3\mathrm{L}$	$p6 - pp2\mathrm{H}$
9	$pp8$	$p4, pp3\mathrm{H}$	$p3 + pp3\mathrm{L}$	$p5, pp3\mathrm{H}$	$p4 - pp3\mathrm{L}$	$p6, pp3\mathrm{H}$	$p5 - pp3\mathrm{L}$	$p7, pp3\mathrm{H}$	$p6 + pp3\mathrm{L}$
10	–	$p1, pp4\mathrm{L}$	$p4 + pp3\mathrm{H}$	$p3, pp4\mathrm{L}$	$p5 - pp3\mathrm{H}$	$p2, pp4\mathrm{H}$	$p6 - pp3\mathrm{H}$	$p4, pp4\mathrm{H}$	$p7 + pp3\mathrm{H}$
11	–	$p2, pp5\mathrm{L}$	$p1 + pp4\mathrm{L}$	$p3, pp5\mathrm{L}$	$p3 - pp4\mathrm{L}$	–	$p2 + pp4\mathrm{H}$	–	$p4 - pp4\mathrm{H}$
12	–	$p3, pp5\mathrm{H}$	$p2 + pp5\mathrm{L}$	$p4, pp5\mathrm{H}$	$p3 - pp5\mathrm{L}$	–	–	–	–
13	–	$p3, pp6\mathrm{L}$	$p3 + pp5\mathrm{H}$	$p4, pp6\mathrm{L}$	$p4 - pp5\mathrm{H}$	–	–	–	–
14	–	$p4, pp6\mathrm{H}$	$p3 - pp6\mathrm{L}$	$p5, pp6\mathrm{H}$	$p4 + pp6\mathrm{L}$	–	–	–	–
15	–	$p3, pp7\mathrm{L}$	$p4 - pp6\mathrm{H}$	$p5, pp7\mathrm{L}$	$p5 + pp6\mathrm{H}$	$p4, pp7\mathrm{H}$	–	$p6, pp7\mathrm{H}$	–
16	–	$p3, pp8\mathrm{L}$	$p3 - pp7\mathrm{L}$	$p4, pp8\mathrm{H}$	$p5 + pp7\mathrm{L}$	–	$p4 - pp7\mathrm{H}$	–	$p6 + pp7\mathrm{H}$
17	–	–	$p3 + pp8\mathrm{L}$	–	$p4 + pp8\mathrm{H}$	–	–	–	–

Library. Dedicated I/O ports are provided for each of the adder-subtracters. The number of entry is estimated to be 18 at most.

3.2.3 Carry Propagator Module (CP)

The module CP consisting of an adder and a register (Carry Register) propagates the carry from lower order coefficients to the adjacent higher order coefficients. The structure of CP module is shown in Fig.6.

It stores carry bits from a coefficient and adds the sign-extended bits to the coefficient input at the next cycle except for the least significant coefficient to which it adds zero. It repeats doing the operation up to the most significant coefficient $p7$.

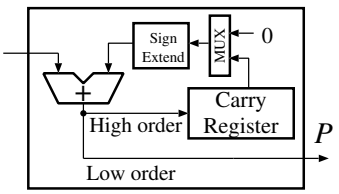

Figure 6. Structure of CP which propagates carries to obtain final results of multiplication.

3.2.4 Controller Module (CTRL)

The module CTRL provides control signals for PPG, ACC, and CP modules at appropriate timings. Typical control signals include addresses for reading and writing PPG and ACC buffers, write enable signals, add/subtract signals for adder-subtracters in ACC, and a signal for the multiplexer to select between carry bits from the carry register and zero.

Under the control of CTRL, PPG performs operations scheduled in the order as shown in Table 2. The table shows the timing when inputs to the multiplier and two adders are given and when their outputs are available in PPG.

The operation sequences for ACC are shown similarly in Table 3. Inputs $pp0$ to $pp8$ in the table corresponds to nine partial products calculated by PPG. Suffices L and H to partial products denote the lower n-bit and the remaining higher parts of the partial products, respectively. Concerning CP, we only need to control the multiplexer to select zero or carry bits to be propagated.

Operations of ACC and CP can start before those of PPG and ACC have completed: Operands of ACC are available at the eighth step of PPG calculation, and from that time ACC can start its operation. Carries are propagated from $p0$ to $p7$ among which $p3$ is the lowest significant coefficient of those taking the longest steps of 17 to calculate as indicated by Table 2. Four significant coefficients higher than $p3$ can be obtained in the following four steps. Therefore, the product P can be obtained by 27=(8-1)+(17-1)+4 steps after multiplier A and multiplicand B are input to this design of R2IKM.

4 Implementation Results and Discussions

The iterative Karatsuba multiplier presented in the previous section was described in hardware language Verilog-HDL. The description was synthesized using Design Compiler Ver. 2004.12-SP2 (Synopsys Inc.). For the synthesis we used CMOS 0.18μm technology cell library developed by VDEC (VLSI Design and Education Center) [13] based on Hitachi's specifications. We put delay-preferred constraints to the synthesizer.

The evaluation results of the design with respect to time, area, and power consumption are shown in Table 4. The basic bit length was varied from 4- to 128-bits. The first and second rows show the basic bit length and the length of multiplier and multiplicand (four times longer than the basic bit length). Time in the fifth row shows operation time (27 $\times$ Delay) required for a multiplication. Internal bit lengths of PPG and ACC are 2- and 5-bit longer than that of basic bit length, respectively, taking the carry into account.

We compare the performance and energy consumption of hardware and software implementation of Karatsuba multiplication. For software we employed exflib Ver.20050709[14] which provides a high-performance Karatsuba multiplication routine. The experimental conditions are Pentium 4 1.7 GHz CPU with the same design rule as our hardware implementation, 512MB main memory, FreeBSD 5.4 OS, and gcc 3.4.2 compiler. The results of performance of R2IKM are shown in Fig.7, where operation time of exflib multiplication is compared with that of R2IKM. In the figure x- and y-axes represent logarithms of multiplication bit length and time required for a multiplication, respectively. The experimental values of exflib multiplication time are well approximated by a line with slope of 1.58 standing for multiplication time of $O(n^{1.58})$ by Karatsuba algorithm.

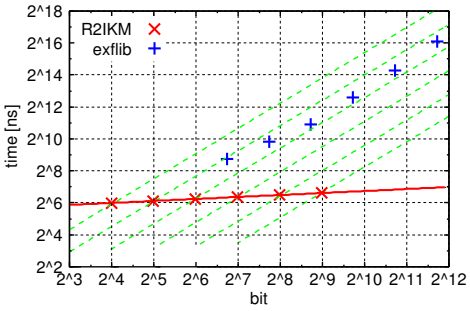

Figure 7. Performance of R2IKM ($\times$) compared with operation time of multiplication by exflib (+). Dashed lines with the slope of 1.58 stand for multiplication time of $O(n^{1.58})$ by Karatsuba algorithm.

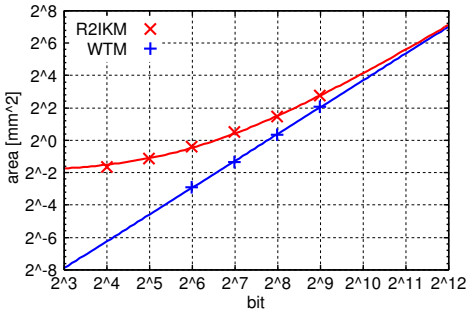

Figure 8. Relation between the area cost and bit length of R2IKM. A point (x, y) denoted by $\times$ represents area cost y of R2IKM with bit length x. Points denoted by + below those by $\times$ indicate area costs of WTM.

We focus our attention to the dashed lines with slope of 1.58 passing through the experimental points of R2IKM of various basic bit lengths. These dashed lines indicate ideal performance of IKM when the recursion times are increased with the bit length of the fundamental WTM modules fixed. The performance ratio of exflib and IKM with WTM of fixed bit length is thus almost constant. The ratio can be increased by increasing the WTM bit length. For instance the performance of R2IKM with 128-bit WTM is approximately 30 times better than that of exflib multiplication. From Table 4 the area of R2IKM with 128-bit WTM is found to be 6.91mm^2 which can be justified depending on applications. The performance of R2IKM with 32-bit WTM is approximately 5 times better than that of exflib multiplication with a small area cost of 1.43mm^2.

From Table 4, energy required for R2IKM with 128-bit WTM to multiply 512-bit numbers is found to be only 151nW·s(=1530mW$\times$98.55ns). On the other hand, it takes 1938ns for Pentium 4 1.7GHz processor consuming 63W[15] to multiply 427-bit (128-digit) numbers by exflib, leading to energy consumption of 122mW·s(=63W$\times$1938ns) which is 807 times larger than that of 128-bit R2IKM with 32-bit WTM.

Table 4. Evaluation results of R2IKM with respect to time, area, and power consumption.

Basic bit length	4	8	16	32	64	128
Multiplication bit length	16	32	64	128	256	512
Delay [ns]	2.36	2.61	2.85	3.08	3.36	3.65
Area [mm^2]	0.32	0.46	0.75	1.43	2.81	6.91
Time [ns]	63.72	70.47	76.95	83.16	90.72	98.55
Power [mW] (1.8V)	34.09	58.60	97.88	220.4	529.7	1530

Figure 8 shows the relation between the area cost and bit length of R2IKM multiplier obtained by performing logic synthesis with delay-preferred constraints . A point (x, y) denoted by $\times$ represents area cost y of R2IKM with bit length x. Points denoted by + below those by $\times$ indicate area costs of WTM synthesized with delay-preferred constraint. A point (x, y) denoted by + represents area cost y of WTM with bit length $x/4$ used within IKM of bit length x. The area cost of WTM approaches $O(n^{1.65})$ as seen from the figure, which differs from $O(n^2)$, the theoretical space complexity of WTM. The difference is caused by giving delay-preffered constraints when synthesizing WTMs. The area cost of R2IKM consists mainly of those of WTM, PPG Buffer, ACC Buffer, and adders. When increasing the basic bit length with the number of recursions unchanged, the length of PPG and ACC Buffer entries increases linearly while their numbers are unchanged. The area costs of PPG and ACC Buffers are thus $O(n)$. The order of area costs of adders is obviously $O(n)$. Therefore, R2IKM has the area cost of WTM $+O(n)$. Consequently, the area of R2IKM approaches to $O(n^{1.65})$.

5 Conclusions

Hardware design of multi-digit integer multiplication of ranging from 16 to more than 512 bits was described and its VLSI realization was evaluated in terms of the cost and performance. The results were compared with those of a software implementation of Karatsuba method. Karatsuba hardware with 128-bit WTM can perform 512-bit multiplications 30 times faster than software at the area cost of 6.91mm^2. Computation energy was found to be nearly 10^{-3} of that consumed by general purpose processor which executes the software version . The results obtained by this study will help system designers for applications requiring multi-digit multiplication to select design alternatives including ASIC realization.

6 Acknowledgments

This research was conducted in collaboration with Synopsys Inc. and Cadence Design System Inc. through the VLSI Design and Education Center, University of Tokyo. It was partially supported by JSPS Grant-in-Aid for Scientific Research (C)(2) 18500048.

References

[1] H. Fujiwara, High-accurate numerical method for integral equations of the first kind under multiple-precision arithmetic, *Theoretical and Applied Mechanics Japan*, Vol. 52, 2003, 193-203.

[2] J. C. Sprott, Chaos and time-series analysis, *Oxford University Press*, 2003.

[3] M. Agrawal, N. Kayal, and N. Saxena, PRIMES is in P, http://www.cse.iitk.ac.in/, 2002.

[4] B. W.-K. Ling, C. Y.-F. Ho and P. K.-S. Tam, Chaotic filter bank for computer cryptography, *Chaos, Solitons & Fractals, In Press*, Available online 5 Jun. 2006.

[5] T.-I Chien and T.-L. Liao, Design of secure digital communication systems using chaotic modulation, cryptography and chaotic synchronization, *Chaos, Solitons and Fractals*, Vol.24, 2005, 241-255.

[6] A. Karatsuba and Y. Ofman, Multiplication of multidigit numbers on automata, *Sov. Phys. Dokl.*, Vol.7, 1963, 595-596.

[7] D. E. Knuth, *The art of computer programming 2nd edition: Seminumerical algorithms*, Vol. 2, Addison-Wesley, MA, 1981.

[8] S. Yazaki and K. Abe, An optimum design of FFT multidigit multiplier and its VLSI implementation, *Bulletin of the University of Electro-Communications*, Vol.18, Nos.1 and 2, Jan. 2006, 39-45.

[9] C. Grabbe, M. Bednara, J. Teich, J. von zur Gathen, and J. Shokrollahi, FPGA designs of parallel high performance GF(2^{233}) multiplier, *Proc. of the IEEE International Symposium on Circuits and Systems*, May 2003, 362-369.

[10] Z. Dyka and P. Langendoerfer, Area efficient hardware implementation of elliptic curve cryptography by iteratively applying Karatsuba's method, *Proc. of the Design, Automation and Test in Europe Conference and Exhibition*, Vol.3, Mar. 2005, 70-75.

[11] M. Shibaoka, N. Takagi, and K. Takagi, Reduced area parallel multiplier based on Karatsuba algorithm, *IEICE General Conference*, Vol.A-3, Mar. 2005, 66. (in Japanese)

[12] S. Yazaki and K. Abe, VLSI Implementation of Karatsuba Algorithm and Its Evaluation, *Proc. The International Workshop on Modern Science and Technology 2006*, Wuhan, China, May 2006, 378-383.

[13] VLSI Design and Education Center Homepage, http://www.vdec.u-tokyo.ac.jp

[14] exflib - Extended Precision Float-Point Arithmetic Library, http://www-an.acs.i.kyoto-u.ac.jp/ fujiwara/ exflib/exflib-index.html

[15] Intel(R) Pentium(R) 4 processor specifications, http:// www.intel.co.jp/products/processor/pentium4/specs.htm