VLSI Implementation of Karatsuba Algorithm and Its Evaluation

Syunji Yazaki[1] and Kôki Abe[1]
[1]Department of Computer Science, The University of Electro-Communications,
Chofu-shi, 182-8585 Japan.

ABSTRACT

VLSI implementation of Karatsuba algorithm for multi-digit multiplication was investigated. We designed 32-bit recursive Karatsuba multiplier (RKM) and found that its critical path delay and area cost are 9.44ns and 0.228mm^2, respectively. Next we designed and evaluated RKM of larger bits. For bit length less than 2^9, Wallace tree multiplier (WTM) has less area cost than RKM, while RKM has less area cost than WTM for bit length larger than 2^9. The area cost of multiplier of 2^9 bits is approximately 30mm^2. Critical path delay of RKM is always larger than that of WTM. We should use WTM instead of RKM as combinational circuits for iterative Karatsuba multiplier (IKM) to have better cost performance.

keywords:multi-digit multiplier, Karatsuba, VLSI

1 INTRODUCTION

Multi-digit arithmetic is used by various applications in recent years, including numerical calculation[1], chaos arithmetic[2], high-performance primality testing[3] and cryptography. For multi-digit multiplication which is frequently used and requires much calculation time, sophisticated algorithms have been proposed such as Karatsuba method($O(n^{1.58})$)[4], FFT method($O(n \log n \log \log n)$)[5] etc., where n stands for bit length. Karatsuba algorithm is employed in multiplication of hundreds to thousands bits, whereas FFT algorithm is used for millions bit multiplication.

Hardware implementation of multi-digit multiplication algorithms is effective for realizing high performance arithmetic. For small digits, VLSI implementation of multiplication such as Wallace tree multiplier has been widely used. However, studies on VLSI implementation of Karatsuba and FFT multiplication algorithms are few. We previously reported results of VLSI implementation of FFT multiplier[6]. In this paper we describe VLSI implementation of Karatsuba multiplier and evaluate its performance and area cost.

This paper is organized as follows: Section 2 briefly explains Karatsuba algorithm and presents two types of its implementations: recursive and iterative Karatsuba multipliers (RKM and IKM). Section 3 describes related work. Section 4 presents design and implementation of 32-bit RKM using Wallace tree multiplier (WTM) of 2^4 bits as unit component. In Section 5 we design RKM of 2^k bits ($4 < k$), evaluate its critical path delay and area cost, and compare them with those of WTM of 2^k bits. Discussions on which multiplier RKM or WTM is used for combinational circuits for IKM are also presented. Finally we present some concluding remarks and future work in Section 6.

2 KARATSUBA ALGORITHM

Let a and b be $2n$-bit integers, $n > 0$. We split them into halves as $a = a_{\mathrm{H}}2^n + a_{\mathrm{L}}$ and $b = b_{\mathrm{H}}2^n + b_{\mathrm{L}}$. Product p of a and b is given by

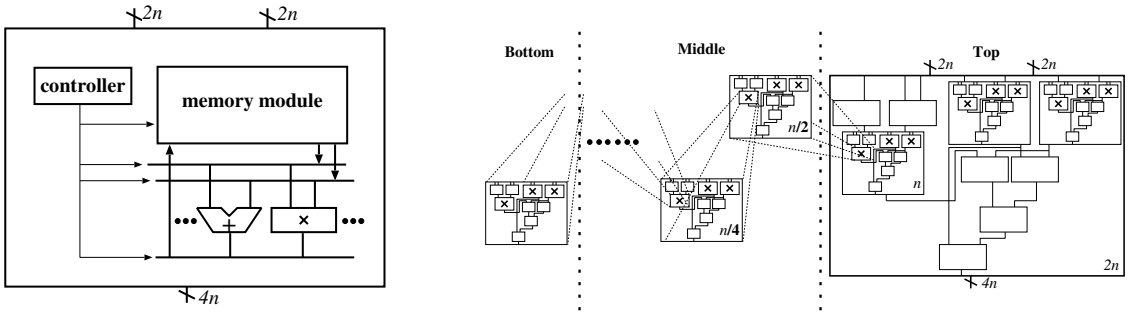
$$p = a \cdot b = (a_{\mathrm{H}}2^n + a_{\mathrm{L}}) \cdot (b_{\mathrm{H}}2^n + b_{\mathrm{L}}) = a_{\mathrm{H}} \cdot b_{\mathrm{H}}2^{2n} + (a_{\mathrm{H}} \cdot b_{\mathrm{L}} + a_{\mathrm{L}} \cdot b_{\mathrm{H}})2^n + a_{\mathrm{L}} \cdot b_{\mathrm{L}}. \quad (1)$$

Using the relation $a_{\mathrm{H}} \cdot b_{\mathrm{L}} + a_{\mathrm{L}} \cdot b_{\mathrm{H}} = (a_{\mathrm{H}} + a_{\mathrm{L}}) \cdot (b_{\mathrm{H}} + b_{\mathrm{L}}) - (a_{\mathrm{H}} \cdot b_{\mathrm{H}} + a_{\mathrm{L}} \cdot b_{\mathrm{L}})$, we can rewrite eq.(1) as follows:

$$p = a_{\mathrm{H}} \cdot b_{\mathrm{H}}2^{2n} + ((a_{\mathrm{H}} + a_{\mathrm{L}}) \cdot (b_{\mathrm{H}} + b_{\mathrm{L}}) - (a_{\mathrm{H}} \cdot b_{\mathrm{H}} + a_{\mathrm{L}} \cdot b_{\mathrm{L}}))2^n + a_{\mathrm{L}} \cdot b_{\mathrm{L}}. \quad (2)$$

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, eq.(2) is applied recursively up to multiplication 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 a sequential circuit as shown in Fig.1(a). We call this architecture as iterative Karatsuba multiplier (IKM). In the recursive approach the algorithm is realized by combinational circuit as shown in 1(b). 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.



(a) Iterative Karatsuba multiplier (IKM).

(b) Recursive Karatsuba multiplier (RKM).

Figure 1: Structures of Karatsuba multiplier.

Multiplier for integers of longer than thousands bits must be realized by IKM because the circuit size would become huge if RKM were used. Multiplier with smaller bit length can be realized by combinational circuit, RKM or WTM. Critical path delay of WTM is obviously less than RKM for any bit length. Concerning with area cost, however, employing RKM with $O(n^{1.58})$ would be better than WTM with $O(n^2)$ for the combinational multiplier. In practice, low order parts and the coefficients of the complexity expression may affect the area costs, as well. Therefore, actual area cost of RKM and WTM needs to be estimated based on implementation result.

In addition we should choose between RKM and WTM to use as combinational multipliers for IKM, depending on the budget allowed. Experimental comparison of area cost and critical path delay between RKM and WTM is required for the design choice. In this paper we implement RKM and compare the results with WTM with respect to their critical path delay and area cost.

3 RELATED WORK

32-bit RKM was implemented by Shibaoka et al.[7]. RKM in [7] is based on two ideas. One is to exclude a CPA (Carry Propagation Adder) at the output stage of Wallace tree and to use CSAs (Carry Save Adders) for addition at intermediate stages. Another is to obtain the binary product by adding 3 or 2 bit addends and augends in such a sequence that those determined earlier are added earlier. Multi-digit Karatsuba multiplier, however, is not dealt with in [7].

Many studies have been reported for applying Karatsuba algorithm to multiplication over Galois field (GF) [8, 9]. Critical path delay and area cost of these GF Karatsuba multipliers can not be compared with integer multiplier because calculations over GF have less computational complexity, although design approaches such as iterative [8] and recursive multipliers can also be applied to integer multiplier.

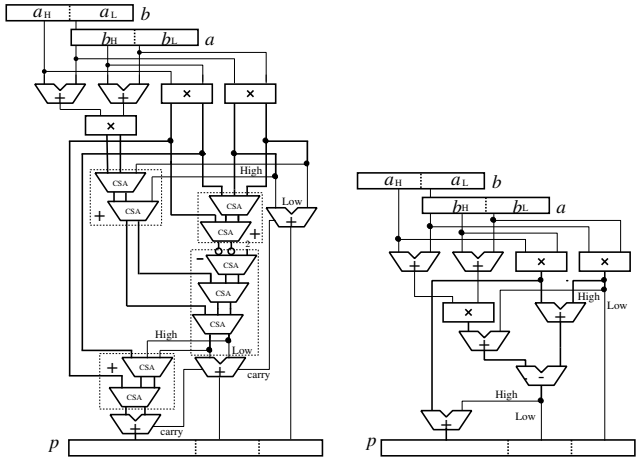
4 DESIGN OF 32-BIT RECURSIVE KARATSUBA MULTIPLIER

4.1 Design

In this section we investigate the simplest RKM which applies Karatsuba algorithm one time. Here we design 32-bit RKM using 16-bit WTM. Structure of the RKM we designed is shown in Fig.2(a). We employed CSAs for addition at intermediate stages and used CPAs only at the output stage of RKM as in [7]. We denote the structure with Carry-Save RKM.

Carry-Save RKM we designed differs from [7] in that we add partial products produced by CSAs using 16- or 32-bit CPAs provided by Synopsys Inc. as an IP (Interectual Property). Synthesizing the Carry-Save RKM description with the IP CPAs we can generate automatically various circuits by varying constrains of critical path delay and area cost. The synthesis result will help us design optimum circuit by providing various alternatives when budget is given.

For comparison, we design another version of RKM that uses CPA for all additions. We denote this version with Binary RKM. Structure of Binary RKM is shown in 2(b).



(a) Carry-Save RKM (b) Binary RKM

Figure 2: Structures of 32-bit RKM.

Table 1: Synthesis Results of 32-bit RKM.

(a) Using HITACHI 0.18μm technology with area-preferred constraint

	delay[ns]	area[mm^2]
Binary RKM	12.03	0.216
Carry-Save RKM	9.44	0.228

(b) Using ROHM 0.35μm technology with delay-preferred constraint

	delay[ns]	area[mm^2]
Carry-Save RKM	8.95	0.741
Ref.[7]	7.70	0.580

4.2 Implementation Results and Discussion

We evaluated the critical path delay and area cost of RKMs designed in previous section. We described design in Verilog-HDL and performed logic synthesis using Design Compiler (Synopsis Inc., Ver. W-2004.12-SP2). For the synthesis we used CMOS 0.18μm technology cell library developed by VDEC (VLSI Design and Education Center) [10] based on Hitachi's specifications. We put area-preferred constraint to Design Compiler to reduce area cost. We also used CMOS 0.35μm technology cell library based on ROHM's specifications which was used in [7]. We put delay-preferred constraint to Design Compiler in this case. Synthesis results are shown in Table 1.

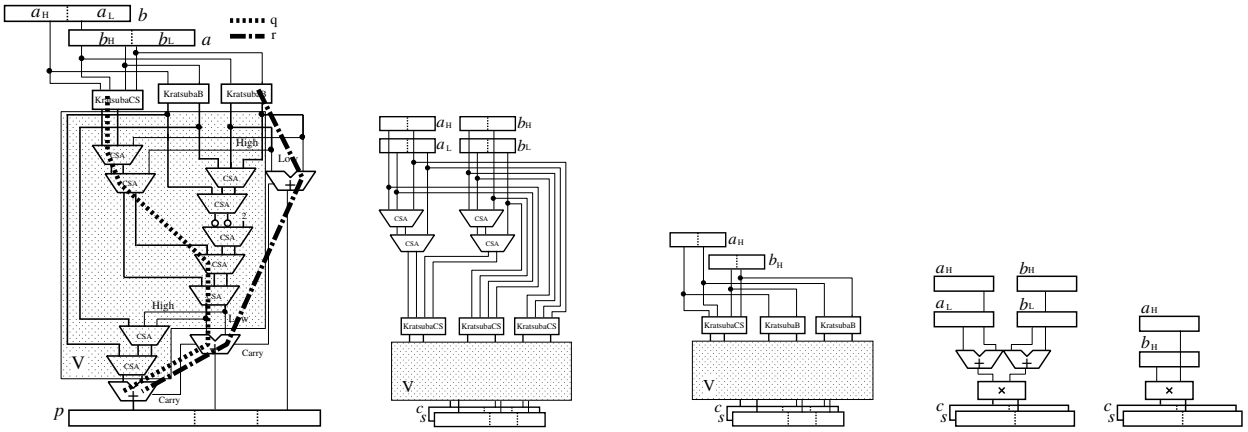
Table 1(a) shows that performing all additions by CSA reduces critical path delay by 22% with area cost increased by 6%. Area cost is increased because more number of full adders are required for carry-save additions than binary additions. Although motivation to use RKM is to reduce the area cost and increased area cost does not justify the motivation, we employ Carry-Save RKM because of its good cost performance ratio. Table 1(b) shows that critical path delay and area cost of Carry-Save RKM are 1.16 times and 1.28 times larger than those in [7], respectively. One of the reason is that CPA was not fully optimized in logic synthesis.

5 DESIGN OF KARATSUBA MULTIPLIER OF LARGER BIT LENGTH

5.1 Design

In this section we design RKM of $2n = 2^k$ bits using unit WTM of 2^l bits, where l and k ($l < k$) are nonnegative integers. Note that previous section dealt with a special case $k = 5$ and $l = 4$. In designing RKM of larger bits, there are choices of what bit length of Wallace tree is to be used for unit multipliers.

We need three kinds of components: (1) Topmost component (T) which needs CPAs to obtain final result as a binary representation. (2) Intermediate components (M) which require no CPAs. (3) Undermost components (B) which include 2^l-bit WTMs with binary inputs and CPAs to convert carry-save to binary representation. In addition, the components M and B have two versions: one with inputs in carry-save representation (M_{cs} and B_{cs}), the other with inputs in binary representation (M_b and B_b). Designs of the modules are shown in Fig.3. KaratsubaCS and KaratsubaB in Fig.3 are modules to be replaced with other modules according to Table2 depending on the number of recursions. In case of one-time recursion KaratsubaCS and KaratsubaB are replaced, respectively, with B_{cs} and B_b in Fig.3(d) and Fig.3(e),



(a) T (b) M_{cs} (c) M_b (d) B_{cs} (e) B_b

Figure 3: Structures of Karatsuba multiplier for larger bits: Topmost component (T), intermediate components (M), and undermost components (B). Sufficies cs and b denote components with inputs in carry-save and binary representations, respectively.

Table 2: Modules to be replaced in recursion.

number of recursions	1	≥ 2
KaratsubaCS	W_{cs}	M_{cs}
KaratsubaB	W_b	M_b

resulting in structure Fig.2(a) obtained in the previous section. In case of more than two-time recursions KaratsubaCS and KaratsubaB are replaced, respectively, with M_{cs} and M_b in Fig.3(b) and Fig.3(c). Module V in Fig.3 performs carry-save addition in Karatsuba algorithm eq.(2).

Now we derive equations to estimate critical path delay and area cost of RKM. Let $DT(k)$ and $AT(k)$ be critical path delay and area cost, respectively, of 2^k-bit RKM, $DM(j)$ and $AM(j)$, $l < j < k$, be those of intermediate RKM, and $DB(l)$ and $AB(l)$ be those of undermost RKM. $DT^{\rm q}(k)$ and $DT^{\rm r}(k)$ are delays along paths q and r, respectively, in Fig.3(a). $D_{\rm CSA}(i)$ and $D_{\rm CPA}(k)$ are critical path delays of 2^i-bit CSA and 2^k-bit CPA, respectively, and $A_{\rm CSA}(i)$ and $A_{\rm CPA}(k)$ are area costs of 2^i-bit CSA and 2^k-bit CPA, respectively, where $i > 0$ is an integer. $D_{\rm W}(l)$ and $A_{\rm W}(l)$ are critical path delays and area costs of 2^l-bit WTM, respectively. From Fig.3(a) we get for critical path delay and area cost as follows.

$$DT(k) = \mathbf{MAX}(DT^{\rm q}(k), DT^{\rm r}(k)), \tag{3}$$
$$DT^{\rm q}(k) = DM_{\rm cs}(k-1) + 4 \cdot D_{\rm CSA}(k) + D_{\rm CPA}(k-1) + D_{\rm CPA}(k), \tag{4}$$
$$DT^{\rm r}(k) = DM_{\rm b}(k-1) + 2 \cdot D_{\rm CPA}(k-1) + D_{\rm CPA}(k), \tag{5}$$
$$DM_{\rm cs}(j) = DM_{\rm cs}(j-1) + 2 \cdot D_{\rm CSA}(j-1) + 6 \cdot D_{\rm CSA}(j), \tag{6}$$
$$DM_{\rm b}(j) = DM_{\rm cs}(j-1) + 6 \cdot D_{\rm CSA}(j), \tag{7}$$
$$DB_{\rm cs}(l) = D_{\rm CPA}(l) + D_{\rm W}(l), \tag{8}$$
$$DB_{\rm b}(l) = D_{\rm W}(l), \tag{9}$$

$$AT(k) = AM_{\rm cs}(k-1) + 2 \cdot AM_{\rm b}(k-1) + 9 \cdot A_{\rm CSA}(k) + 2 \cdot A_{\rm CPA}(k-1) + \cdot A_{\rm CPA}(k), \tag{10}$$
$$AM_{\rm cs}(j) = 3 \cdot AM_{\rm cs}(j-1) + 4 \cdot A_{\rm CSA}(j-1) + 9 \cdot A_{\rm CSA}(j), \tag{11}$$
$$AM_{\rm b}(j) = AM_{\rm cs}(j-1) + 2 \cdot AM_{\rm b}(j-1) + 9 \cdot A_{\rm CSA}(j), \tag{12}$$
$$AB_{\rm cs}(l) = 2 \cdot A_{\rm CPA}(l) + A_{\rm W}(l), \tag{13}$$
$$AB_{\rm b}(l) = A_{\rm W}(l). \tag{14}$$

In case $j - 1 = l$, for critical path delay and area cost of intermediate RKM, those of undermost components are used. That is, $DM_{\rm cs}(l) = DB_{\rm cs}(l)$, and so on.

Table 3: Critical path delay and area cost of CPA (DW01_add).

bit length	2^4	2^5	2^6	2^7	2^8	2^9
delay [ns]	1.38	1.82	2.31	3.02	3.97	5.21
area [mm^2]	0.006	0.012	0.026	0.057	0.126	0.275

Table 4: Critical path delay and area cost of RKM and WTM.

(a) RKM

bit length	2^5	2^6	2^7	2^8	2^9
delay [ns]	9.81	13.37	16.95	21.21	26.24
area [mm^2]	0.270	0.972	3.275	10.602	33.469

(b) WTM

bit length	2^4	2^5	2^6	2^7
delay [ns]	4.15	5.44	6.90	8.09
area [mm^2]	0.054	0.184	0.675	2.574

5.2 Implementation Results and Discussion

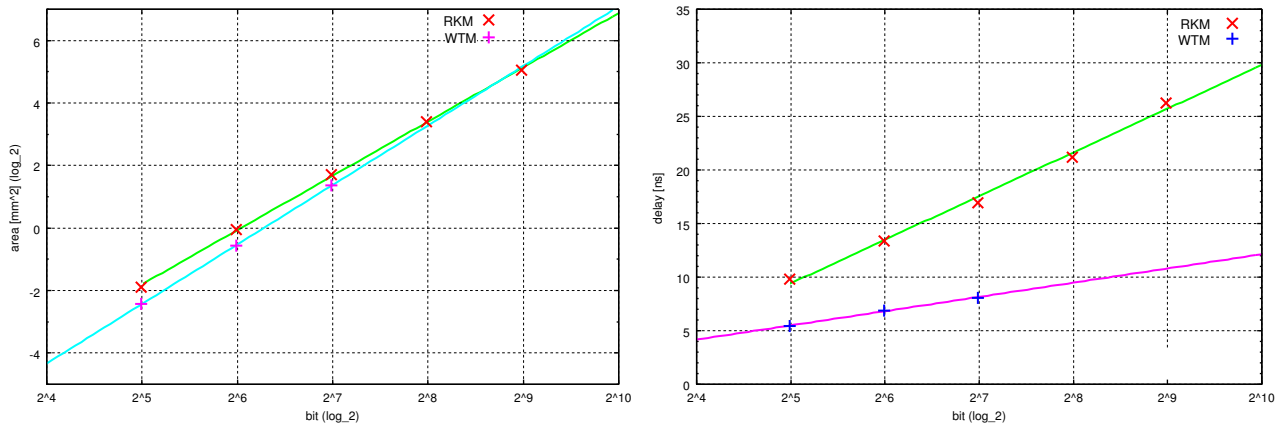
Using equations (3) to (14), we evaluate critical path delay and area cost of RKM of up to $2^9 = 512$ bit length. Here we use 16-bit WTM as its fundamental components ($l = 4$) with $D_{\mathrm{W}}(2^4) = 4.15$ns, $A_{\mathrm{W}}(2^4) = 0.054$mm^2.

Critical path delay and area cost of CPA were obtained by performing logic synthesis of adder DW01_add included in Design Ware Library provided by Synopsys Inc., and are shown in Table 3. In performing logic synthesis, we specified the condition so as to use CLA (Carry Lookahead Adder) and put area-prefered constraint. Critical path delay and area cost of CSA were 0.27ns and 0.004mm^2 in case of 16-bit length. Critical path delay of CSA remains unchanged as increasing bit length while the area cost is doubled by doubling bit length.

The critical path delay and area cost of RKM of bit length ranging from 2^5 to 2^9 were estimated using the parameters presented above. The results are shown in Table 4(a). Table 4(b) shows those of WTM which was synthesized using DW02_mult in Design Ware Library. Fig.4(a) compares the results by plotting area costs of RKM and WTM with x-axis of $\log_2(bit\ length)$ and y-axis of $\log_2(area)$. Fig.4(b) compares the results by plotting critical path delays of RKM and WTM with x-axis of $\log_2(bit\ length)$ and y-axis of $delay$.

The critical path delay and area cost of the synthesized RKM of 2^6 bits were 14.14ns and 0.864mm^2, respectively, compared with the estimated values 13.37ns and 0.972mm^2. The difference between estimated values shown in Table 4 and synthesis results is attributed to optimization by logic synthesizer and considered to be reasonable. We note that for RKM $\log_2(area)$ as a function of $\log_2(bit\ length)$ is approximated to be a line with slope of 1.73. Difference from the theoritical value $O(n^{1.58})$ is attributed to lower order complexity term of additions in Karatsuba algorithm. From Fig.4(a) area complexity of WTM is found to be $O(n^{1.90})$ which is close to theoretical value $O(n^2)$. For bit length less than 2^9 WTM has less area cost than RKM while RKM has less area cost than WTM for larger bit length than 2^9. The area cost of multipliers of 2^9 bits is approximately 30mm^2.

From Fig.4(b) it is confirmed that critical path delay of both RKM and WTM is $O(\log n)$. However, the slope of RKM is larger than that of WTM. For speed-prefered design, WTM is better to use as combinational multipliers for IKM.



(a) Area cost

(b) Critical path delay

Figure 4: Comparison between RKM and WTM with respect to area cost and critical path delay.

For area-prefered design, using RKM as combinational multipliers for IKM is not so effective considering that the area cost at 2^9 bit length, above which RKM becomes better, is 30mm^2 that is larger enough in most cases. Consequently, we should use WTM istead of RKM as combinational multipliers for iterative Karatsuba multiplier to have better cost performance. Using what bit length of WTM as combinational multiplier for IKM depends on the budget allowed: using WTM of longer bit length will result in faster and larger IKM.

6 CONCLUSIONS AND FUTUREWORK

In this paper, we investigated which kind of combinational multipliers, Wallace tree multiplier (WTM) and recursive Karatsuba multiplier (RKM), should be used for iterative Karatsuba multiplier (IKM). First we designed 32-bit RKM and evaluated critical path delay and area cost. We found that 32-bit RKM could be realized with critical path delay of 9.44ns and area cost of 0.228 mm^2. Next we designed RKM of larger bits and estimated its critical path delay and area cost. Results show that for bit length less than 2^9 WTM has less area cost than RKM while RKM has less area cost than WTM for bit length larger than 2^9. The area cost of multipliers of 2^9 bit length is approximately 30mm^2. Considering that critical path delay of RKM is always larger than that of WTM, we should use WTM istead of RKM as combinational multipliers for iterative Karatsuba multiplier to have better cost performance. Design and evaluation of IKM and its comparison with software Karatsuba multiplication belong to future work.

ACKNOWLEDGMENT

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) 1650026.

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, pp.193-203, 2003.

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

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

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

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

[8] 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 pp.70-75, Mar. 2005.

[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, pp.362-369, May 2003.

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