Scalable Unified Dual-Radix Architecture for Montgomery Multiplication in $GF(P)$ and $GF(2^n)$

Kazuyuki TANIMURA, Ryuta NARA, Shunitsu KOHARA, Kazunori SHIMIZU,
Youhua SHI, Nozomu TOGAWA, Masao YANAGISAWA, Tatsuo OHTSUKI

Dept. of Computer Science
Waseda University
Shinjuku, Tokyo 169–8555, Japan
Tel: +81–3–3209–3211(5716)
Fax: +81–3–3204–4875
e-mail: tanimura@yanagi.comm.waseda.ac.jp

Abstract— Modular multiplication is the most dominant arithmetic operation in elliptic curve cryptography (ECC), which is a type of public-key cryptography. Montgomery multiplication is commonly used as a technique for the modular multiplication and required scalability since the bit length of operands varies depending on the security levels. Also, ECC is performed in $GF(P)$ or $GF(2^n)$, and unified architectures for $GF(P)$ and $GF(2^n)$ multiplier are needed. However, in previous works, changing frequency or dual-radix architecture is necessary to deal with delay-time difference between $GF(P)$ and $GF(2^n)$ circuits of the multiplier because the critical path of $GF(P)$ circuit is longer. This paper proposes a scalable unified dual-radix architecture for Montgomery multiplication in $GF(P)$ and $GF(2^n)$. The proposed architecture unifies 4 parallel radix-2^{16} multipliers in $GF(P)$ and a radix-2^{64} multiplier in $GF(2^n)$ into a single unit. Applying lower radix to $GF(P)$ multiplier shortens its critical path and makes it possible to compute the operands in the two fields using the same multiplier at the same frequency so that clock dividers to deal with the delay-time difference are not required. Moreover, parallel architecture in $GF(P)$ reduces the clock cycles increased by dual-radix approach. Consequently, the proposed architecture achieves to compute $GF(P)$ 256-bit Montgomery multiplication in $0.23\mu s$.

I. Introduction

Elliptic curve cryptography (ECC) is a type of public-key cryptography. Currently, the de-facto standard of public-key cryptography is RSA; however, 1024-bit or longer key is recommended for RSA in order to establish secure communication. Because the security of ECC with 160-bit key is competitive with that of RSA with 1024-bit key, hardware implementation of ECC can be more efficient in terms of area and operation time.

Montgomery multiplication[4] is known as a technique for modular multiplication. The advantage of using Montgomery multiplication is that it enables to avoid divisions by the modulus after multiplications. Since the modular multiplication is the most dominant arithmetic operation in ECC, Montgomery multiplication is a common approach to shorten the encryption and decryption time.

ECC is performed in either of two finite fields: prime field $GF(P)$ or binary extension field $GF(2^n)$. Because ECDSA[1] that is the standardized elliptic curve-based signature scheme uses both fields, Montgomery multipliers need to be practicable in both fields as well. The difference between multipliers in $GF(P)$ and $GF(2^n)$ is their delay times. In $GF(2^n)$, carries generated in additions are not propagated so that the critical path of $GF(2^n)$ circuit is shorter than that of $GF(P)$ circuit. In other words, addition in $GF(2^n)$ is defined as XOR.

The bit length of operands for Montgomery multiplier varies from 160bits to 1024bits or more depending on required security level. Implementing individual multipliers for each bit-length of input on a single chip makes the hardware cost expensive. On the other hand, breaking that long operand into certain bit-length digits and calculating them with a single arithmetic unit with iteration can be scalable and reduce hardware area. This is the reason why scalable algorithms and architectures are preferable especially when hardware resources are restricted.

In previous works such as [2, 3, 5–10], several scalable Montgomery multipliers were proposed. Ref.[2, 5–7] proposed unified architecture of $GF(P)$ and $GF(2^n)$ circuit whereas [3, 8–10] are dedicated to $GF(P)$. The difficulty putting $GF(P)$ and $GF(2^n)$ circuits together is how to deal with the delay-time difference.

Ref.[2, 6] proposed architectures with radix-2 multipliers. When k is defined as the bit length of the digits for breaking n-bit operands into m digits ($n = k \cdot m$), radix is defined as 2^k, and there is a trade-off between circuit delay time and total clock cycles by the radix size. The multipliers in [2, 6] include both $GF(P)$ and $GF(2^n)$ circuits so that it can compute Montgomery multiplication in both fields. However, the clock cycles of Montgomery multiplication increases proportionally to the square of m, where m is the number of digits. Thus, it is difficult to enhance throughput with these radix-2 multipliers.

Ref.[5] proposed architecture with 64-bit$\times 64$-bit multiplier, and it produces high throughput. As far as we know, this is the fastest design in unified multipliers because of the high-radix architecture, but this makes delay-time difference between $GF(P)$ and $GF(2^n)$ circuit larger. Then the frequency of $GF(P)$ mode is constrained at $137.7MHz$ meanwhile $GF(2^n)$ mode can be driven at $510.2MHz$. Moreover, the

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

multiplier has to be capable of adopting different frequencies.

To deal with the problem of the delay-time difference between $GF(P)$ and $GF(2^n)$ circuits, [7] proposed dual-radix architecture. In general, lower-radix makes critical path shorter. Ref.[7] reported that applying radix-2 for $GF(P)$ and radix-2^2 for $GF(2^n)$ could reduce the delay-time difference. However, this method increases $GF(P)$ clock cycles dramatically because the radix of $GF(P)$ multiplier have to be small. As a result, the Montgomery multiplication time of this method is longer than that of [5]. Moreover, this proposal is applicable only when the size of radix in $GF(P)$ is small, such as 2^1, 2^2, or 2^3.

This paper proposes a scalable unified dual-radix architecture for Montgomery multiplication in $GF(P)$ and $GF(2^n)$. The proposed architecture unifies 4 parallel radix-2^{16} multipliers in $GF(P)$ and a radix-2^{64} multiplier in $GF(2^n)$ into a single unit. Additionally, this architecture reduces the clock cycles increased by dual-radix approach with the parallel multiplier in $GF(P)$, and most $GF(P)$ and $GF(2^n)$ circuits can be shared. Applying lower radix to $GF(P)$ multiplier shortens its critical path and makes it possible to compute the numbers in the two fields using the same multiplier. Therefore, the proposed Montgomery multiplier can be driven at same frequency in both $GF(P)$ and $GF(2^n)$, consequently, the total operation time of our proposal becomes shorter than that of any previous works.

The rest of this paper is organized as follows; section II introduces the Montgomery multiplication algorithms; section III proposes an architecture for a scalable and dual-radix unified Montgomery multiplier which reduces the delay time difference between the $GF(P)$ and $GF(2^n)$ circuit; section IV reports the implementation results and comparisons with previous works. Finally, conclusions are given in section V.

II. Montgomery Multiplication

Montgomery multiplication is a common approach for modular multiplication, and the modular multiplication is the most dominant arithmetic operation in ECC. In this section, we introduce algorithms for the Montgomery multiplication.

A. Montgomery Multiplication in $GF(P)$

Fig.1 shows the fundamental Montgomery multiplication algorithm in $GF(P)$. In Fig.1, k represents the bit-length of the digits as breaking n-bit operands into m digits ($n = k \cdot m$). Besides, capital alphabets represent n-bit polynomials, and small alphabets represent single digit. Other notations we use in this paper are listed in TableI.

Montgomery multiplication algorithm (Fig.1) avoids modulus operation by adding the product of A and B to multiples of P. Lower $n$bits of the result is guaranteed to become zeros. q must be pre-calculated. Fig.2 shows the Montgomery multiplication where $m = 4$. In Fig.2, C is the output of Montgomery multiplication.

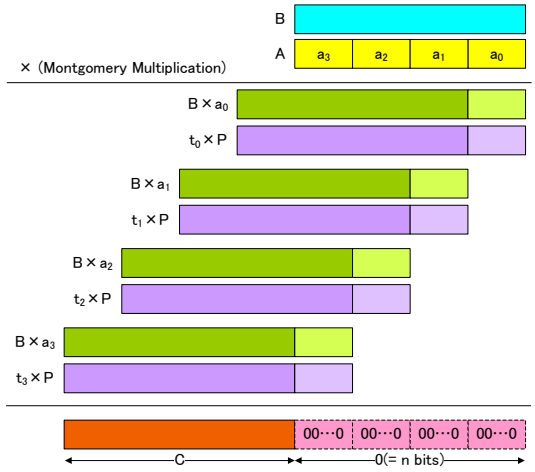

Fig. 2. Montgomery Multiplication.

TABLE I
Notation

Parameter	Meaning
A, B	Operands for Montgomery multiplication
a_i, b_i	k-bit (w-bit) digit (word) of A, B at position i
P	Modulus for Montgomery multiplication
q	$q = P^{-1} \bmod r$
S	Partial products in the multiplication process
C	Multiplication result
n	Bit length of A, B , and P
r	Radix ($r = 2^k$)
k	Bit length of the digit
m	Number of digits ($m = \lceil n/k \rceil$)
w	Bit length of the word
e	Number of words ($e = \lceil n/w \rceil$)

INPUT: $A = (a_{m-1}, \cdots, a_1, a_0)_{2^k}$, $B, P\ (0 \le A, B < P)$
INPUT: $q = -P^{-1} \bmod 2^k$
OUTPUT: $C = AB2^{-n} \bmod P$

$C := 0$
for $i = 0$ to $m - 1$
  $t_i := (c_0 + a_i b_0)\, q \bmod 2^k$
  $C := (c_i + a_i B + t_i P) \big/ 2^k$
if $(C > P)$ then $C := C - P$

Fig. 1. Montgomery Multiplication in $GF(P)$.

B. Montgomery Multiplication in $GF(2^n)$

Montgomery multiplication in $GF(2^n)$ is executed with the almost same steps as in $GF(P)$. Fig.3 shows the fundamental Montgomery algorithm in $GF(2^n)$. However, unlike

INPUT: $A(x) = (a_{m-1}, \cdots, a_1, a_0)_{x^k}$, $B(x)$, $P(x)$
INPUT: $q(x) = P(x)^{-1} \bmod x^k$
OUTPUT: $C(x) = A(x)\, B(x)\, x^{-n} \bmod P(x)$

$C(x) := 0$
for $i = 0$ to $m - 1$
  $t_i(x) := [c_0(x) + a_i(x)\, b_0(x)]\, q(x) \bmod x^k$
  $C(x) := [c_i(x) + a_i(x)\, B + t_i(x)\, P(x)] \big/ x^k$

Fig. 3. Montgomery Multiplication in $GF(2^n)$.

INPUT: $A = (a_{m-1}, \cdots, a_1, a_0)_{2^k}$
INPUT: $B = (b_{m-1}, \cdots, b_1, b_0)_{2^k}$
INPUT: $P = (p_{m-1}, \cdots, p_1, p_0)_{2^k}$ $(0 \leq A, B < P)$
INPUT: $q = -P^{-1} \bmod 2^k$
OUTPUT: $C = AB2^{-n} \bmod P$

$C := 0$
for $i = 0$ to $m-1$
  $z := 0$
  $t_i := (c_0 + a_i b_0)\, q \bmod 2^k$
  for $j = 0$ to $m-1$
    $S := c_j + a_i b_j + t_i p_j + z$
    if $(j \neq 0)$ then $c_{j-1} := S \bmod 2^k$
    $z := S / 2^k$
  $c_{m-1} := z$
if $(C > P)$ then $C := C - P$

Fig. 4. Scalable Montgomery Multiplication in $GF(P)$.

INPUT: $A(x) = (a_{m-1}, \cdots, a_1, a_0)_{x^k}$
INPUT: $B(x) = (b_{m-1}, \cdots, b_1, b_0)_{x^k}$
INPUT: $P(x) = (p_{m-1}, \cdots, p_1, p_0)_{x^k}$
INPUT: $q(x) = P(x)^{-1} \bmod x^k$
OUTPUT: $C(x) = A(x)\, B(x)\, x^{-n} \bmod P(x)$

$C(x) := 0$
for $i = 0$ to $m-1$
  $z(x) := 0$
  $t_i(x) := [c_0(x) + a_i(x)\, b_0(x)]\, q(x) \bmod x^k$
  for $j = 0$ to $m-1$
    $S(x) := c_j(x) + a_i(x)\, b_j(x) + t_i(x)\, p_j(x) + z(x)$
    if $(j \neq 0)$ then $c_{j-1}(x) := S(x) \bmod x^k$
    $z(x) := S(x) / x^k$
  $c_{m-1}(x) := z(x)$

Fig. 5. Scalable Montgomery Multiplication for $GF(2^n)$.

Fig.1, Fig.3 shows that the final subtraction is not required in $GF(2^n)$.

C. Scalable Montgomery Multiplication

The algorithms shown in Fig.1 and Fig.3 do not have scalability to the change of n. Since the operand A is divided by m into k-bit digits ($n = k \cdot m$), it is possible to correspond to change of n by adjusting m. In contrast, the operand B is not divided, and n-bit×k-bit multiplier is required for Fig.1 and Fig.3 so that it is impossible to correspond to the change of n. On the other hand, the algorithm used in [2, 3, 5–10] has scalability because the operand B is divided by e into w-bit words ($n = w \cdot e$), and w-bit×k-bit multipliers are applied. When n is changed, e is adjusted correspondingly as well as m.

The architecture in [5] comprises a multiplier accumulator (MAC) with four 64-bit inputs and one 128-bit output. In addition, [5] adopted "finely integrated operand scanning method (FIOSM)", which was proposed in [11]. Fig.4 and Fig.5 show the FIOSM in $GF(P)$ and $GF(2^n)$. In [5] 64-bit MAC is used in both $GF(P)$ and $GF(2^n)$. Comparing to $GF(2^n)$ circuit delay time, $GF(P)$ circuit delay time is longer so that the frequency of $GF(P)$ mode should be lower than $137.7MHz$, meanwhile $GF(2^n)$ mode can be driven at $510.2MHz$.

To deal with the problem of the delay-time difference between $GF(P)$ and $GF(2^n)$, [7] proposed dual-radix architecture. In [7], applying radix-2 to $GF(P)$ and radix-2^2 to $GF(2^n)$ circuit reduced the delay-time difference. However, the total Montgomery multiplication time of [7] is longer than that of [5] because [7] simply made the radix for $GF(P)$ smaller than that for $GF(2^n)$, and this requires more clock cycles.

III. Proposed Montgomery Multiplier

A. Delay-Time Difference on Each Radix Size

Fig.6 shows that the delay-time difference between multipliers in $GF(P)$ and $GF(2^n)$. Fig.6 was acquired by synthesizing the multipliers described in VHDL using Synopsys Design Compiler with STARC $90nm$ process library[1]. In Fig.6, the larger k, the greater delay-time difference between multipliers $GF(P)$ and $GF(2^n)$. Reducing the delay time of $GF(P)$ circuit enables to avoid using clock dividers in order to be capable of adopting different frequencies. The delay time of $GF(P)$ multiplier achieves 3.7 times longer than that of $GF(2^n)$ multiplier when $k = 64$.

B. Dual-Radix Unified Multiplier

In this section, we propose a Montgomery multiplier architecture which unifies 4 parallel radix-2^{16} multipliers in $GF(P)$ and a radix-2^{64} multiplier in $GF(2^n)$ into a single unit in order to reduce the delay-time difference between $GF(P)$ and $GF(2^n)$ circuits.

Calculating $GF(P)$ operands, we apply lower radix to $GF(P)$ multiplier to shorten its critical path and delay time. In addition, this parallel architecture in $GF(P)$ reduces the clock cycles increased by dual-radix approach. As Fig.6 shows, the delay time of 64-bit ($k = 64$) multiplier in $GF(P)$ is approximately same as the delay time of 16-bit ($k = 16$) multiplier in $GF(2^n)$. Hence, we chose radix-2^{64} for $GF(2^n)$ and radix-2^{16} for $GF(P)$. Because the MAC has 64-bit input, four 16-bit multipliers in $GF(P)$ can be parallelized. Fig.7 shows the block diagram of the unified multiplier including 4 parallel radix-2^{16} multipliers in $GF(P)$ and the radix-2^{64} multiplier

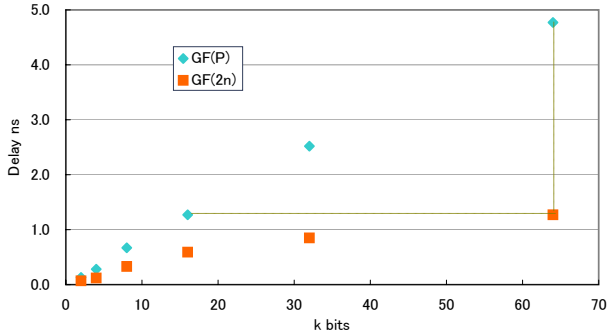

Fig. 6. Delay Time of Multipliers in $GF(P)$ and $GF(2^n)$.

[1]STARC $90nm$ process library is developed in a chip fabrication program provided by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Semiconductor Technology Academic Research Center.

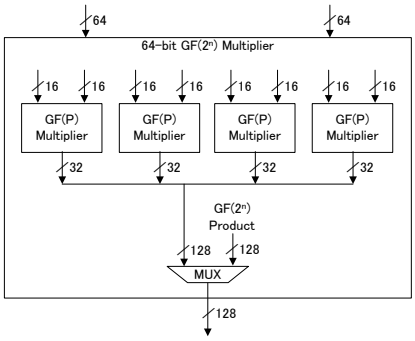

Fig. 7. Dual-Radix Unified Multiplier.

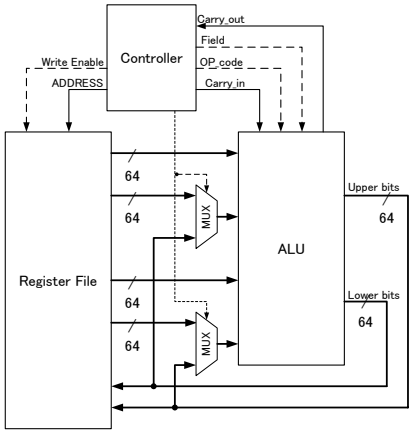

Fig. 8. Scalable Dual-Radix Unified Montgomery Multiplier in $GF(P)$ and $GF(2^n)$.

in $GF(2^n)$. In other words, the multiplier works as 4 individual 16-bit multipliers in $GF(P)$ or single 64-bit multiplier in $GF(2^n)$. Thus the delay-time difference is reduced while high speed processing is enabled at the same frequency in both fields.

C. Architecture of the Proposed Multiplier

Fig.8 shows the block diagram of the proposed Montgomery multiplier. In Fig.8, the ALU consists of MAC, adder-subtracter, and bitwise-XOR circuit which are needed for conducting the algorithms presented in Fig.4 and Fig.5. Additionally, Fig.9 shows the dual-radix unified multiplier placed in the ALU depicted in Fig.8. In Fig.9, 64-bit operands A and B yield a 128-bit product in $GF(2^n)$ or four 32-bit product in $GF(P)$. The control signals from the controller determine in which field the calculation should be performed. MAC operation is completed by adding other 2 operands and the each product. Fig.9 comprises 4 unified multiplier and 12 $GF(2^n)$ multiplier. Fig.10 shows 16-bit unified multiplier in Fig.9. Finally, Fig.11 shows 4-bit unified multiplier in Fig.10.

TABLE II
NUMBER OF CYCLES FOR RADIX-2^{16} MONTGOMERY MULTIPLICATION IN $GF(P)$.

n (bits)	Number of cycles	
	Non-parallel ($NS = 1$)	4 parallel ($NS = 4$)
160	241	78
192	337	83
224	449	132
256	577	165

D. Clock Cycles

The number of clock cycles required for the Montgomery multiplication in $GF(2^n)$ with the proposed architecture is expressed as in Eq.(1), where one MAC operation is done in 1 clock cycle.

$$2m^2 + 3m + 1 \tag{1}$$

Similarly, the number of clock cycles required for the Montgomery multiplication in $GF(P)$ with the proposed architecture is expressed as in Eq.(2), where NS stands for parallel number. Eq.(2) expresses the worst case that the final subtraction in Fig.4 is conducted.

$$\left\lceil \frac{2m^2}{NS} \right\rceil + \left\lceil \frac{2m}{NS} \right\rceil + 4\left((m-1)\,\%NS\right) + m + 1 \tag{2}$$

TableII shows the comparison of required clock cycles between non-parallel multiplier ($NS = 1$) and 4 parallel multipliers ($NS = 4$) with the algorithm in Fig.4. Although the non-parallel radix-2^{16} architecture ($NS = 1$) requires 577 clock cycles to compute 256-bit Montgomery multiplication, the 4 parallel radix-2^{16} architecture ($NS = 4$) requires 3.4 times less the clock cycles: 165 clock cycles.

IV. EXPERIMENTAL RESULTS

We implemented ALU (MAC, adder-subtracter, and bitwise-XOR circuit), register file, and controller depicted in Fig.8 for the proposed scalable dual-radix unified Montgomery multiplier in VHDL, and synthesized them using Synopsys Design Compiler with STARC $90nm$ process library. In this section, we evaluate the operation time and area of the proposed Montgomery multiplier.

A. Operation Time

TableIII shows the comparison of Montgomery multiplication operation time between our proposal and previous works. In TableIII, 256-bit MM time represents 256-bit Montgomery multiplication operation time, and the area size is calculated on the basis of 2-input NAND gate. Also, No. of Stages in TableIII represents how many stages are arrayed in a row, where one stage has one MAC. Since the implementation result indicated that the delay time was $1.4ns$, the proposed Montgomery multiplier can be driven at $714MHz$, theoretically, without considering the wire delay.

In TableIII, the Montgomery multiplication operation time of our proposal is shorter than any other that of previous approach in $GF(2^n)$. The second fastest Montgomery multiplier

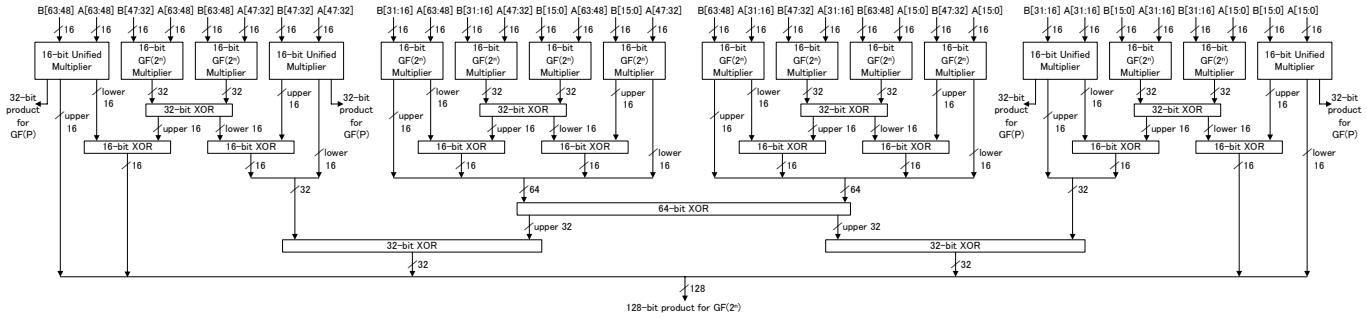

Fig. 9. 64-bit Dual-Radix Unified Multiplier.

in $GF(2^n)$ is the approach of [5], but the Montgomery multiplication operation time of our proposal is as long as that of the approach by [5] in $GF(2^n)$ when they are normalized in frequency. This is because we used the same algorithm and radix as [5] did. TableIV shows the comparison of the required clock cycles for one Montgomery multiplication.

In $GF(P)$, the proposed Montgomery multiplier is the fastest compared with previous works. Although the second fastest Montgomery multiplier in $GF(P)$ proposed in [5] requires 3.4 times less the clock cycles our proposal does (TableIV), the delay time of the $GF(P)$ circuit proposed in [5] is 3.7 times longer than that of the $GF(2^n)$ circuit proposed in [5]. Unlike [5], in this work, the delay time of the $GF(P)$ circuit is as long as that of the $GF(2^n)$ circuit. Therefore, the proposed Montgomery multiplier in $GF(P)$ at $510.2MHz$ is 1.1 times faster in operation time than the multiplier proposed in [5]. More importantly, the proposed architecture allows us to perform both Montgomery multiplications in $GF(P)$ and $GF(2^n)$ at the same frequency so that clock dividers or other additional circuit for the delay-time difference are not required.

The architectures proposed in [3, 8–10] are not unified so that they can not compute both $GF(P)$ and $GF(2^n)$ operands. Furthermore, [2, 6] applied radix-2 architectures, and [7] applied radix-2 in $GF(P)$ and radix-2^2 in $GF(2^n)$, they are relatively slow. In [5], the frequency of $GF(P)$ mode is constrained at $137.7MHz$ meanwhile $GF(2^n)$ mode can be driven at $510.2MHz$, and the design has to be capable of adopting different frequencies.

B. Area

In TableIII, the area of the proposed Montgomery multiplier is the largest compared with previous works except implementation on FPGA. This is because the area of the proposal includes $125kgates$ of the register file depicted in Fig. 8. In contrast, the area described in [6–10] includes only ALU, the area described in [5] includes just ALU and controller; they also need external RAMs to store operands and results. The area of the controller in our proposal was $4kgates$, and the ALU was $27kgates$, and the total was $31kgates$. Hence, the area of the proposed Montgomery multiplier is almost same as that of [6–10] and smaller than that of [5]. TableV shows the area of each unit in our proposal.

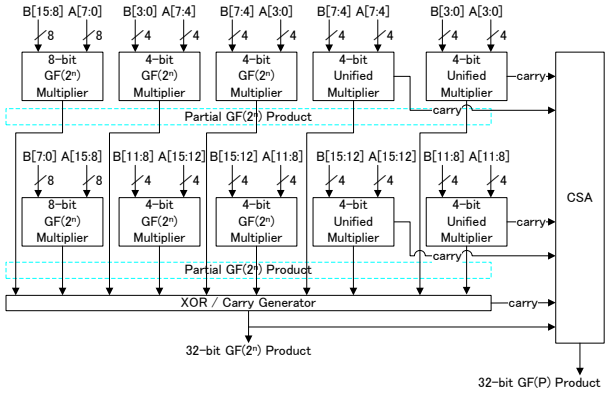

Fig. 10. 16-bit Unified Multiplier.

TABLE V
AREA OF PROPOSED SCALABLE DUAL-RADIX UNIFIED MONTGOMERY MULTIPLIER.

Function Block	Area ($kgates$)
ALU	27
Controller	4
Register File	125

V. SUMMARY AND CONCLUSIONS

This paper proposed a scalable unified dual-radix architecture for Montgomery multiplication in $GF(P)$ and $GF(2^n)$. The proposed architecture unified 4 parallel radix-2^{16} multipliers in $GF(P)$ and a radix-2^{64} multiplier in $GF(2^n)$ into a single unit. Applying lower radix to $GF(P)$ multiplier shortened its critical path and made it possible to compute the numbers in the two fields using the same multiplier. In addition, this architecture reduced the clock cycles increased by dual-radix approach with the parallel multiplier in $GF(P)$, and most $GF(P)$ and $GF(2^n)$ circuits was be able to be shared. Therefore, the proposed Montgomery multiplier was capable of being driven at same frequency in both $GF(P)$ and $GF(2^n)$, consequently, the total operation time of our proposal became shorter than that of any previous works.

TABLE III
COMPARISON OF MONTGOMERY MULTIPLIERS

Reference	Technology	Area	Frequency	Unified	Field	Radix	No. of Stages	256-bit MM time
This work	90 nm CMOS	$156kgates^2$	$714MHz$	Yes	$GF(P)$	2^{16}	$1_{(4\text{parallelized})}$	$0.23\mu s$
This work	90 nm CMOS	$156kgates^2$	$714MHz$	Yes	$GF(2^n)$	2^{64}	1	$63ns$
[2]	Xilinx Virtex Pro	$1514LUTs$	$144MHz$	Yes	$GF(P)$	2	64	$1.1ms$
[2]	Xilinx Virtex Pro	$1514LUTs$	$144MHz$	Yes	$GF(2^n)$	2	64	-
[3]	Xilinx Virtex II	$2593LUTs$	$135MHz$	No	$GF(P)$	2^{16}	16	$200\mu s$
[5]	$0.13\mu m$CMOS	$107kgates$	$137.7MHz$	Yes	$GF(P)$	2^{64}	1	$0.36\mu s$
[5]	$0.13\mu m$CMOS	$107kgates$	$510.2MHz$	Yes	$GF(2^n)$	2^{64}	1	$88ns$
[6]	$1.2\mu m$CMOS	–	$80MHz$	Yes	$GF(P)$	2	2	$6.6\mu s$
[6]	$1.2\mu m$CMOS	–	–	Yes	$GF(2^n)$	2	2	–
[7]	$0.5\mu m$CMOS	$30kgates$	$203MHz$	Yes	$GF(P)$	2	2	$2.6\mu s$
[7]	$0.5\mu m$CMOS	$30kgates$	–	Yes	$GF(2^n)$	2^2	2	–
[8]	$0.5\mu m$CMOS	$28kgates$	$80MHz$	No	$GF(P)$	2	40	$7.4\mu s$
[10]	$0.5\mu m$CMOS	$28kgates$	$64MHz$	No	$GF(P)$	2^3	15	$3.2\mu s$

TABLE IV
NUMBER OF CYCLES FOR MONTGOMERY MULTIPLICATION.

Reference	Radix r		Number of cycles							
			$GF(P)$				$GF(2^n)$			
	$GF(P)$	$GF(2^n)$	$n=160$	$n=192$	$n=224$	$n=256$	$n=160$	$n=192$	$n=224$	$n=256$
This work	2^{16}	2^{64}	78	83	132	165	28	28	45	45
[5]	2^{64}	2^{64}	31	31	49	49	28	28	45	45

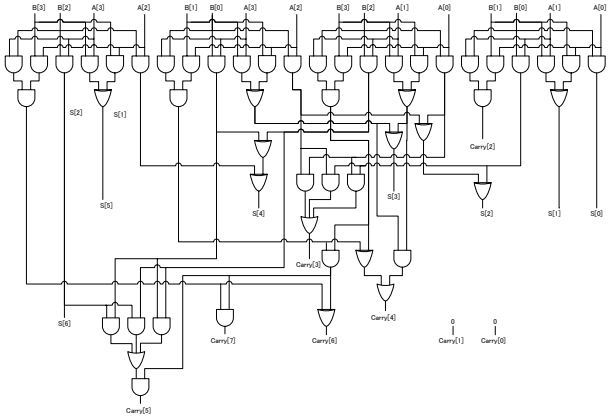

Fig. 11. 4-bit Unified Multiplier.

ACKNOWLEDGEMENTS

This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc. This research was also supported by "Ambient SoC Global COE Program of Waseda University" of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

REFERENCES

[1] "Digital Signature Standard (DSS)," FIPS PUB 186-2, National Institute of Standards and Technology, `http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf`, October 2001.

[2] D. Harris, R. Krishnamurthy, M. Anders, S. Mathew, and S. Hsu, "An improved unified scalable radix-2 Montgomery multiplier," *Proc. of the 17th IEEE Symposium on Computer Arithmetic*, June 2005.

[3] K. Kelley and D. Harris "Parallelized very high radix scalable Montgomery multipliers," *Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers, 2005*, pp.1196-1200, October 2005.

[4] P.L. Montgomery, "Modular multiplication without trial division," *Math. Computing*, Vol.44, No.170, pp.519-521, April 1985.

[5] A. Satoh and K. Takano, "A scalable dual-field elliptic curve cryptographic processor," *IEEE Transactions on Computers*, Vol.52, No.4, pp.449-460, April 2003.

[6] E. Savas, A.F. Tenca, and Ç.K. Koç, "A scalable and unified multiplier architecture for finite fields $GF(p)$ and $GF(2^m)$," *Proc. Second Int'l Workshop Cryptographic Hardware and Embedded Systems-CHES 2000*, pp.277-292, August 2000.

[7] E. Savas, A.F. Tenca, and Ç.K. Koç, "Multiplier architectures for $GF(p)$ and $GF(2^n)$," *IEE Proc. of Computers and Digital Techniques*, Vol.151, No.2, pp.147-160, March 2004.

[8] A.F. Tenca and Ç.K. Koç, "A scalable architecture for modular multiplication based on Montgomery's algorithm," *IEEE Transactions on Computers*, Vol.52, No.9, pp.1215-1221, September 2003.

[9] A.F. Tenca and Ç.K. Koç, "A scalable architecture for Montgomery multiplication," *Proc. First Int'l Workshop Cryptographic Hardware and Embedded Systems-CHES 99*, pp.94-108, August 1999.

[10] A.F. Tenca, G. Todorov, and Ç.K. Koç, "High-radix design of a scalable modular multiplier," *Proc. Workshop Cryptographic Hardware and Embedded Systems-CHES 2001*, pp.185-201, May 2001.

[11] Ç.K. Koç, T. Acar, and B.S. Kaliski, "Analyzing and comparing Montgomery multiplication algorithms," *IEEE Micro*, Vol.16, No.3, pp.26-33, June 1996.

[2] including the register file, while the others not.