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

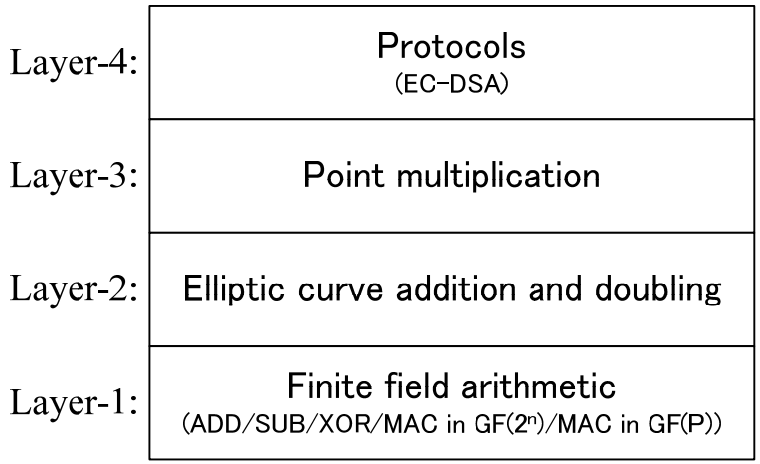
Department of Computer Science,
Waseda University

Outline

- Introduction
- Montgomery Multiplication
- Proposed Montgomery Multiplier
- Experimental Result
- Conclusion

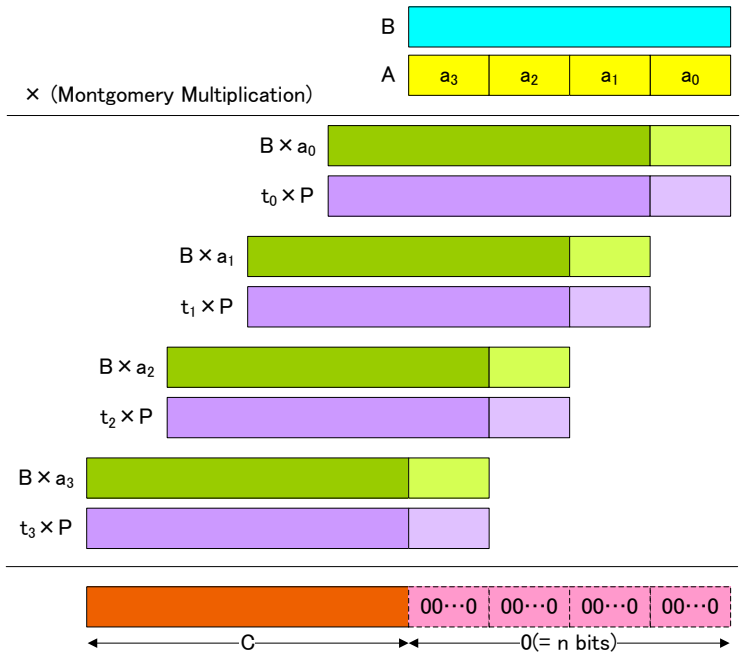
Introduction

- Elliptic Curve Cryptography (ECC)
  - Public-key Cryptography
  - Security level
    - ECC with 160-bit key = RSA with 1024-bit key
  - Area and operation-time efficient hardware can be implemented
- Montgomery multiplication
  - most dominant arithmetic operation



Montgomery Multiplication

- Input: A, B
- Output: $C = A \cdot B \cdot 2^{-n} \bmod P$
  - P: parameter of ECC
  - n: bit width of P

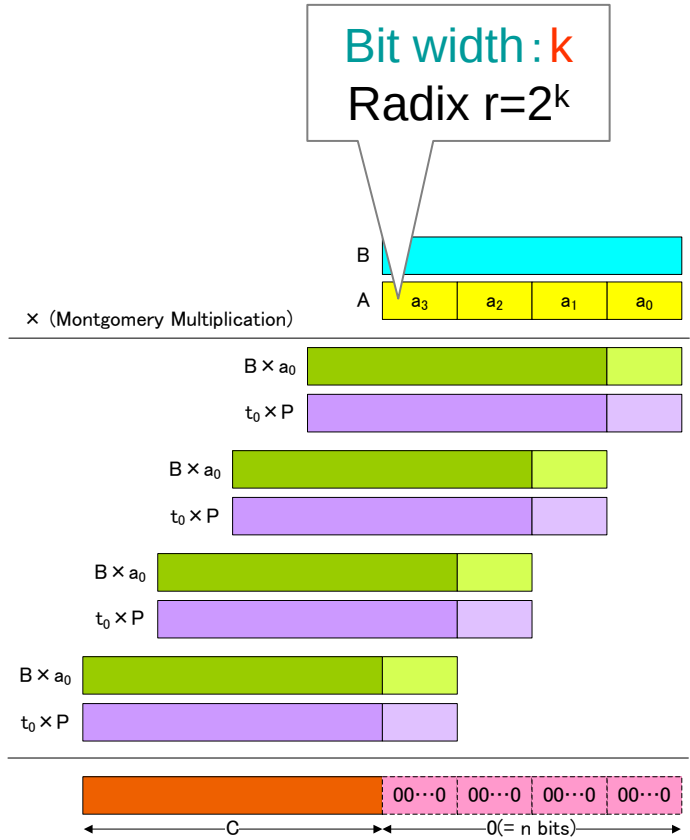


INPUT: $A = (a_{m-1}, \cdots, a_1, a_0)_{2^k}, B, p\,(0 \le A, B < P)$
INPUT: $q = -P^{-1} \bmod 2^k$
OUTPUT: $C = AB2^{-n} \bmod P$

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

Scalable Montgomery Multiplication

- Bit width of operands
  - varies from 160 to 256 bits depending on security levels
- Scalability
  - Divide operands by 64bits
    - Use 64-bit × 64-bit multiplier iteratively
  - Radix-2^{64} architecture



High Radix vs. Low Radix

- Radix($=2^k$)
  - Low radix: 2^1~2^3 [2,4,6,7,8,9,10]
    - Short delay time
    - More clock cycles is required
  - High radix: 2^{16}~2^{64} [3,5]
    - Long delay time
    - Fewer clock cycles is required
- The total operation time of high radix architecture is shorter
    - Decreasing the clock cycles affects the total operation time more than the delay time does

Galois Field

- The operands of Montgomery multiplier are in…
  - GF(P): *Prime field*
    - P is a 160～256-bit prime number
      - E.g. 010b+011b=101b
  - GF(2^n): *Binary extension field*
    - n is a 160～256-bit number
    - Addition is defined as XOR
      - E.g. 010b+011b=001b

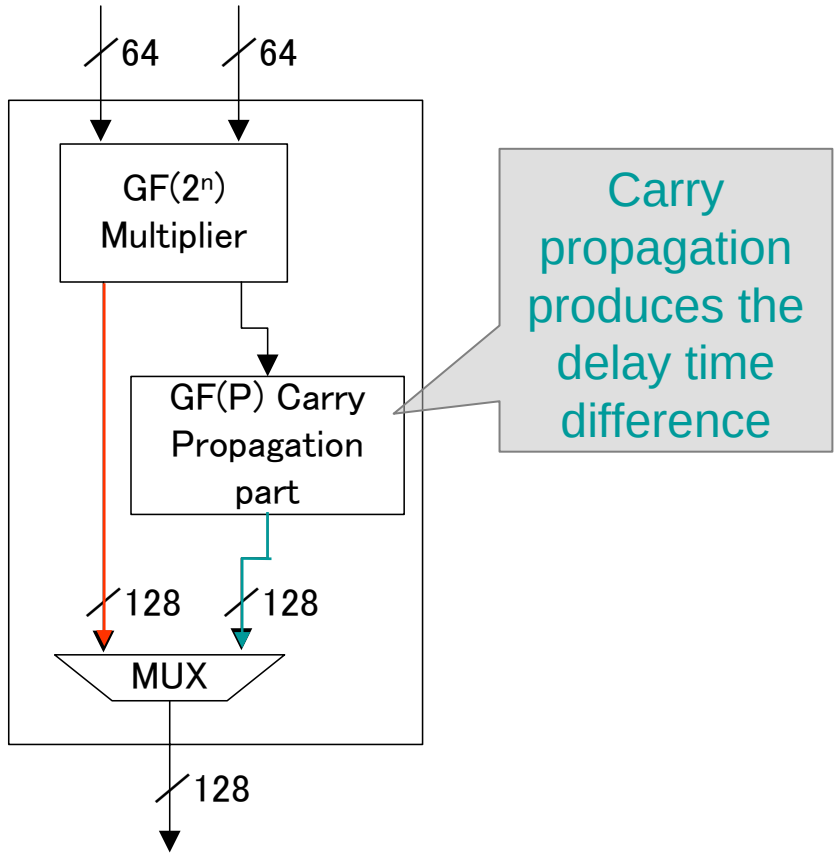
Unified Montgomery Multiplier

- Elliptic curve-based signature scheme EC-DSA[1] is standardized in both GF(P) and $GF(2^n)$ fields
- Unified Montgomery Multiplier [2,5,6,7]:
  - Can compute both GF(P) and $GF(2^n)$ numbers

  - [1] National Institute of Standards and Technology, 2001
  - [2] D. Harris et al., Proc. of the 17th IEEE Symposium on Computer Arithmetic, 2005
  - [5] A.Sato et al., IEEE Transactions on Computers, 2003
  - [6] E. Savas et al., Proc. of 2nd CHES, 2000
  - [7] E. Savas et al., IEE Proc. of Computers and Digital Techniques, 2004

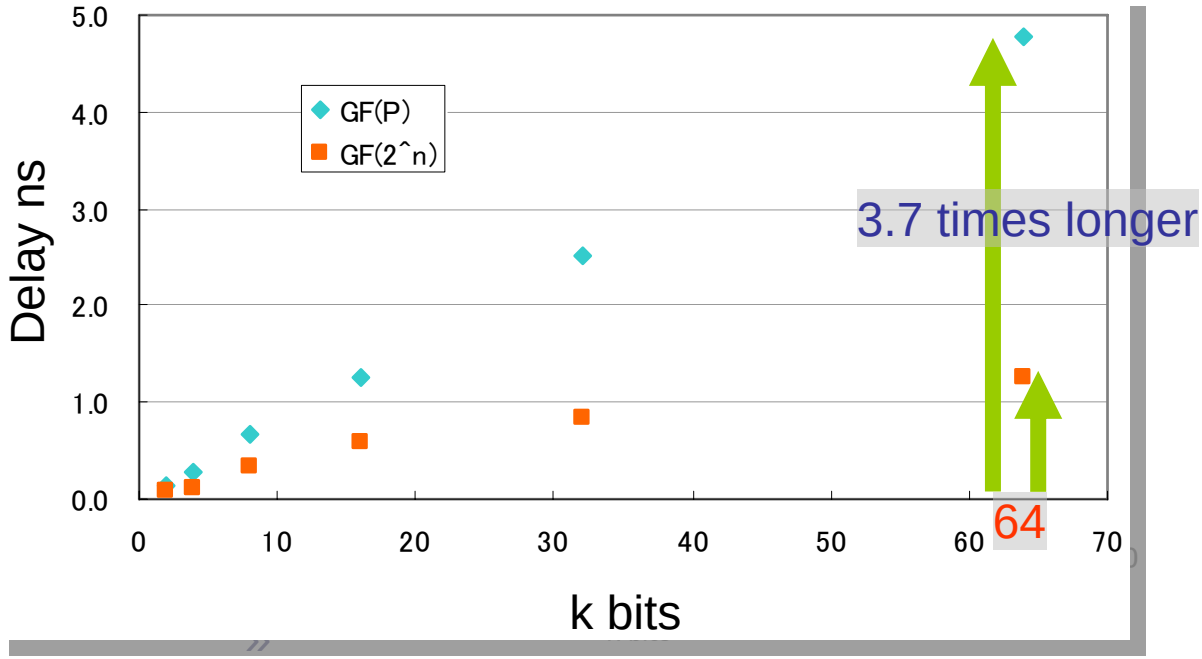
Problem of Unified Architecture

- Delay time difference between GF(P) and GF(2^n) circuits
  - Delay of the circuit
    - GF(P) > GF(2^n)



Delay time difference GF(2^n) and GF(P)

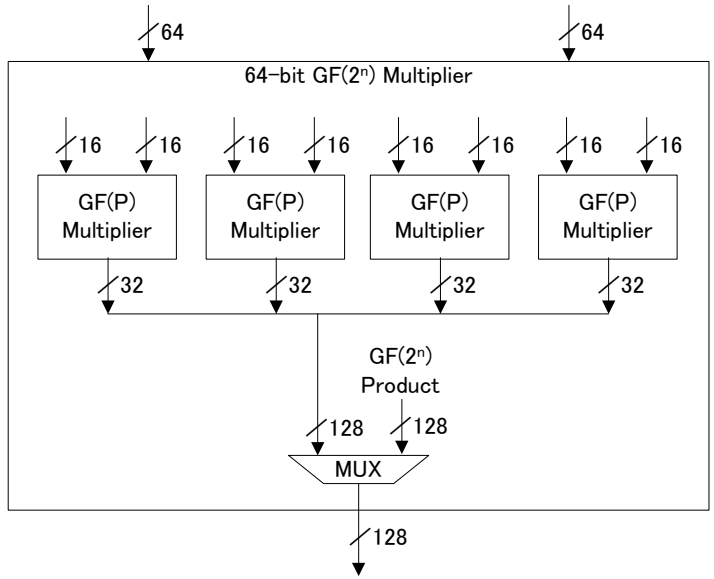
- The larger radix, the longer delay time difference
  - The merit (short delay) of GF(2^n) is ruined in unified architecture



Synthesized by DesignCompiler with STARC90nm process library

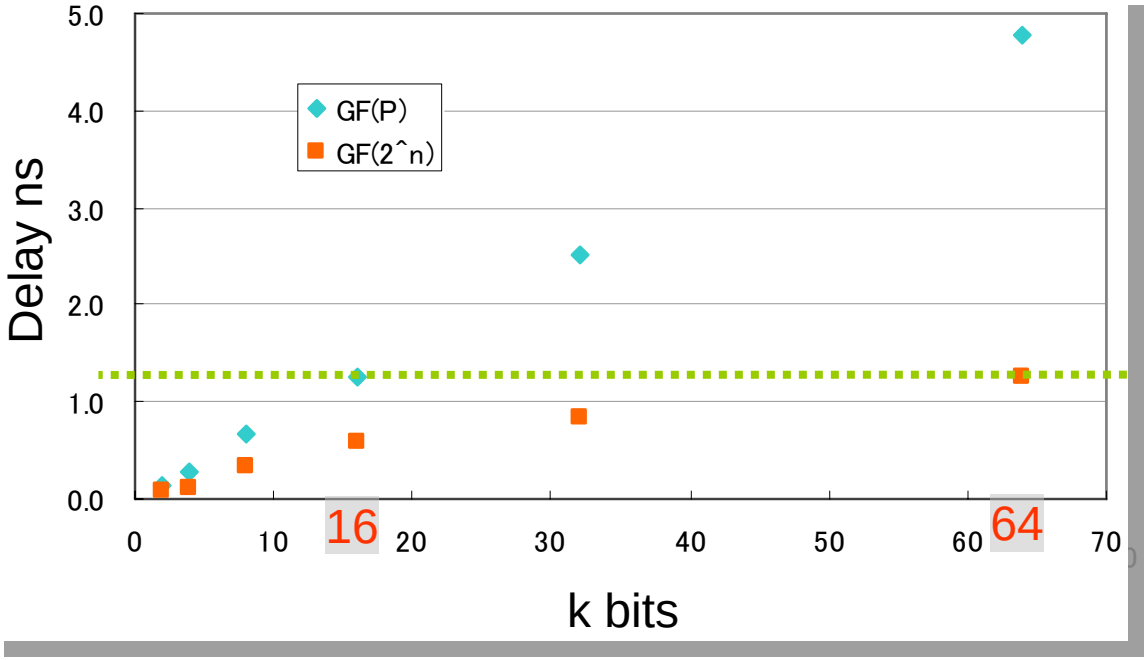
Proposed Montgomery Multiplier

- Scalable
- Unified : GF(P) and GF(2^n)
- Dual-radix
  - GF(P) : radix-2^{16} × 4units
  - GF(2^n) : radix-2^{64}
    - Applying lower radix to GF(P) enables to reduce its delay as long as GF(2^n)
    - Reduce the clock cycles in GF(P) by computing in 4 parallel



Delay time difference between GF(P) and GF(2^n)

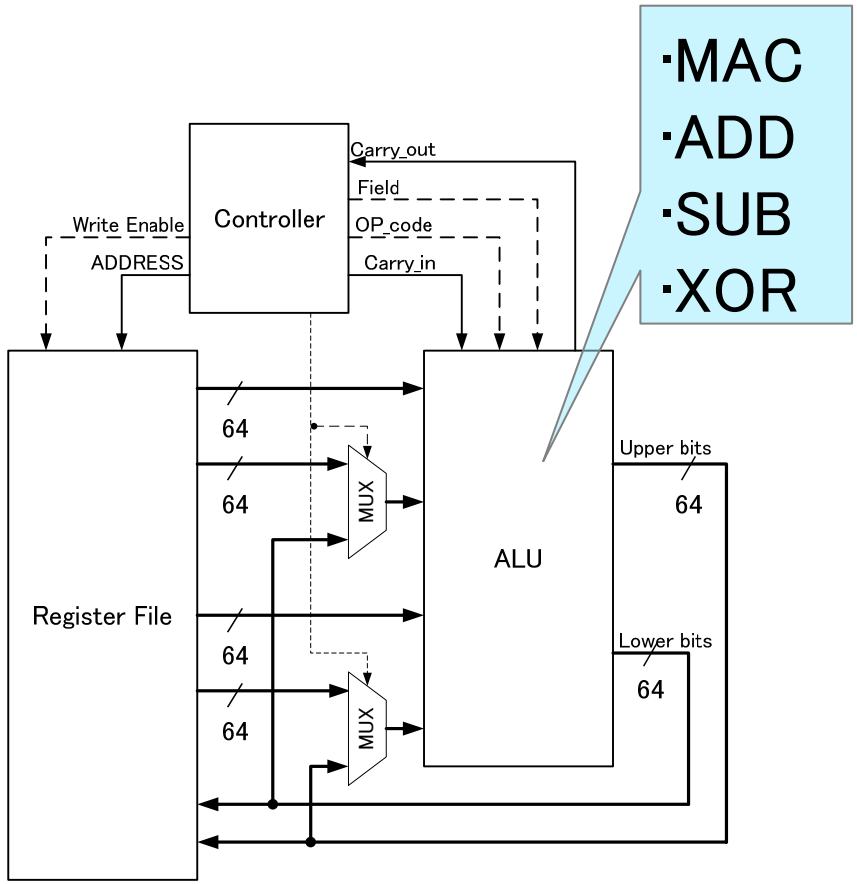
- Delay time of radix-2^{16} multiplier in GF(P) is as long as that of radix-2^{64} multiplier in GF(2^n)



Synthesized by DesignCompiler
with STARC90nm process library

Implementation

- Described in VHDL
- Synthesized using DesignCompiler
- Library
  - STARC90nm process library

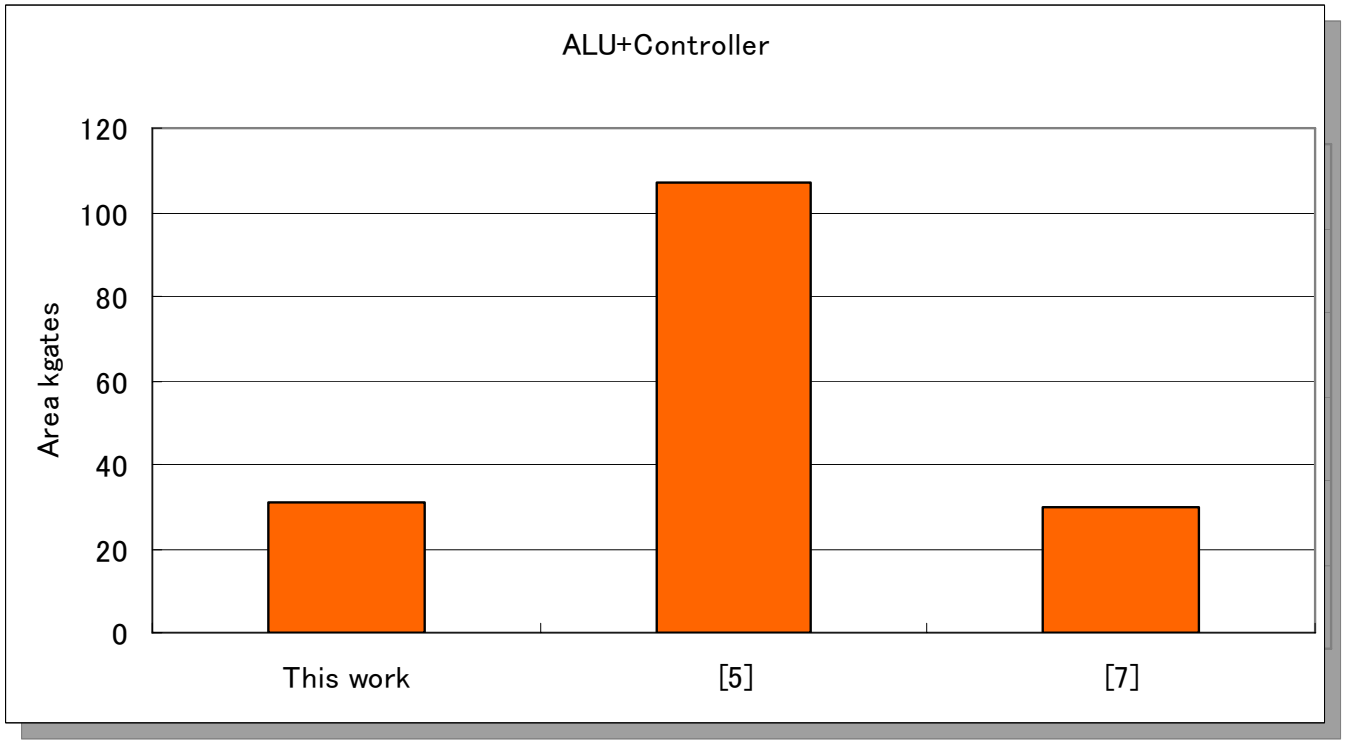


Comparison in Operation Time

Ref.	Tech.	Field	Frequency	Radix	256-bit time
This work	90nm	GF(P)	714 MHz	2^{16}	0.23 μs
		GF(2^n)		2^{64}	63 ns
[5]	0.13 μm	GF(P)	137.7 MHz	2^{64}	0.36 μs
		GF(2^n)	510.2 MHz	2^{64}	88 ns

- Proposed Montgomery multiplier in GF(P) is 11% faster at 510.2MHz
- Dual-radix approach requires more clock cycles, but parallel architecture can cancel them out.
  - [5] A.Sato et al., IEEE Transactions on Computers, 2003

Comparison in Area



- Area of ALU＋Controller is 29% of [5], 103% of [7]
  - [5] A.Sato et al., IEEE Transactions on Computers, 2003
  - [7] E.Savas et al., Computers and Digital Techniques, 2004

Conclusion

- Scalable Unified Dual-Radix Montgomery Multiplier
  - Applying lower radix to GF(P) enables to reduce its delay as long as GF(2^n)
  - Reduce the clock cycles by parallel architecture in GF(P)
  - The area of logic part in proposal is almost same or smaller than other approaches.
  - Can drive both field multipliers at same frequency, which will result in shortening encryption and decryption time

Thank you

Kazuyuki TANIMURA

tanimura@yanagi.comm.waseda.ac.jp

Appendix

Previous Works

- Radix-2^{64} approach in [5] is the fastest Montgomery multiplier

- [5] applied Finely integrated operand scanning method (FIOSM) as an algorithm proposed in[11]

  - [5] A.Sato et al., IEEE Transactions on Computers, 2003
  - [11] C.K.Koc et al. , IEEE Micro, 1996

Scalable Algorithm in GF(P)

- FIOSM[5,11]
  - Double loop
  - Clock cycles increase proportionally to m^2 , where m is the number of digits
    - E.g. 160=5·32
    - n=m·k
      - n: bit width of operands
      - m:number of digits
      - k: bit width of digits

GF(P)

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: $P = (p_{m-1}, \cdots, p_1, p_0)_{2^k}$ $(0 \leqq A, B < P)$
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 $C := C - P$

Scalable Algorithm in GF(2^n)

- FIOSM[5,11]
  - Double loop
  - Clock cycles increase proportionally to m^2
  - Same as GF(P) except final subtraction is not required

GF(2^n)

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

$C(x) := 0$
for $i = 0$ to $m - 1$
  $z(x) := 0$
  $t_i(x) := [c_0(x) + a_i(x) b_0(x)] q(x) \bmod x^r$
    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 $c_{j-1}(x) := S(x) \bmod x^r$
    $z(x) := S(x) / x^r$
  $c_{m-1}(x) := z(x)$

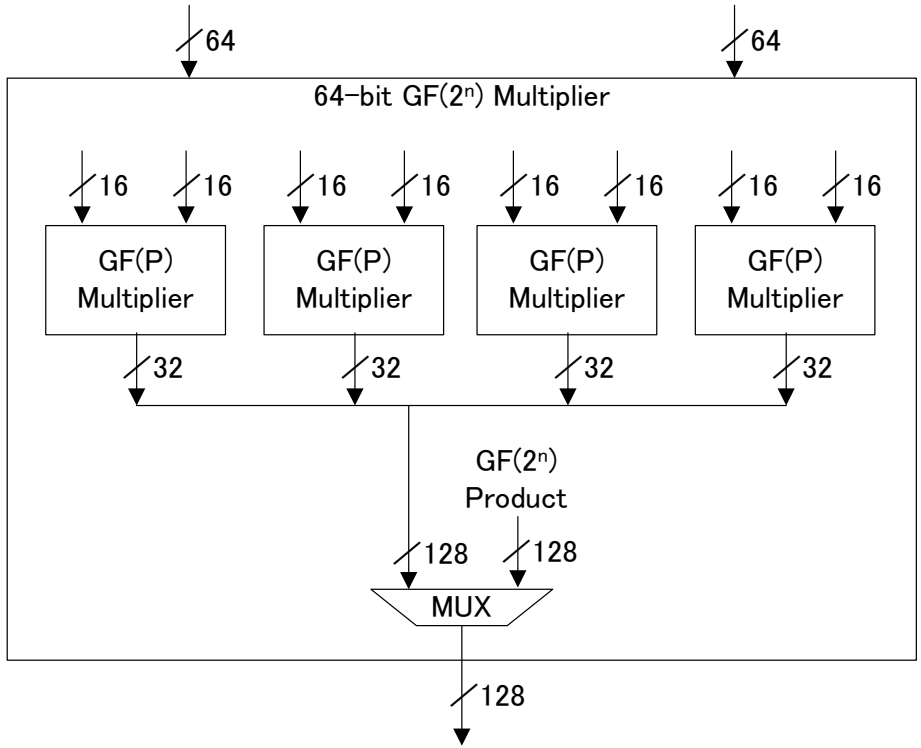
Previous Approaches for Reducing The Delay in GF(P)

- Drive GF(P) circuit slower than $GF(2^n)$[5]
  - Clock dividers are needed to change the field
- Use lower radix for GF(P) and reduce the delay of the GF(P) circut[7]
  - Clock cycles increase dramatically

  - [5] A.Sato et al., IEEE Transactions on Computers, 2003
  - [7] E.Savas et al., Computers and Digital Techniques, 2004

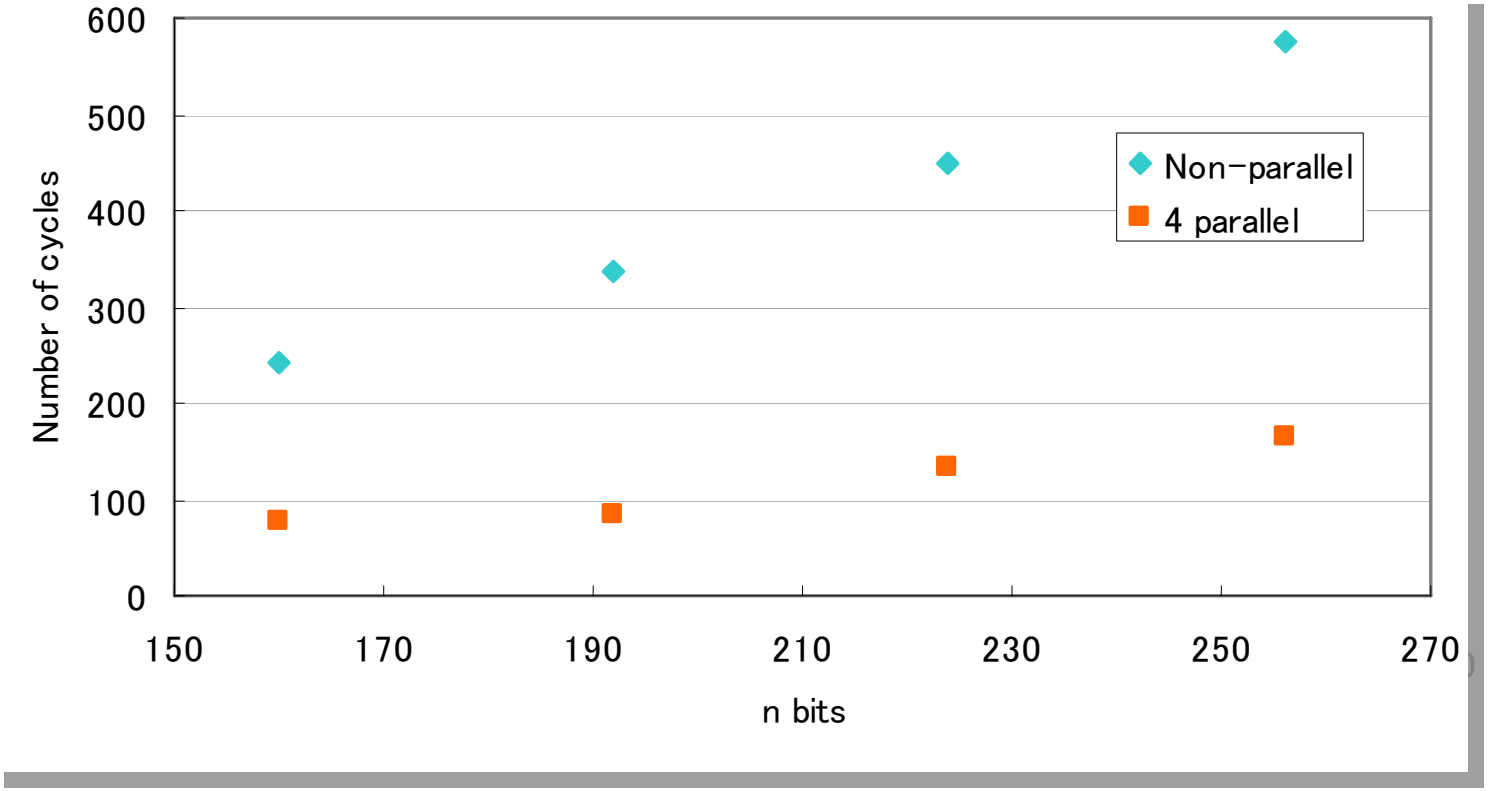
Block Diagram of the Proposal

- 64-bit multiplier in GF(2^n)
- 4 16-bit multipliers in parallel in GF(P)



Reduce Clock Cycles

- Reduce the clock cycles with 4 parallel architecture in GF(P)



Comparison in Clock cycles

Reference	Number of Clock cycles	
	GF(P)	GF(2^n)
This work	165	45
[5]	49	45

- In GF(P), proposal requires 3.4 times more clock cycles than [5] does because [5] applies larger radix
- However, frequency of proposal in GF(P) reaches 3.7 times faster than that of [5], so that proposed Montgomery multiplier is faster
  - [5] A.Sato et al., IEEE Transactions on Computers, 2003

Future Work

- Incorporate the proposed multiplier into the entire cryptographic system
- Implement the multiplier into cryptographic system LSIs