ELectronics EXpress

IEICE Electronics Express, Vol.1, No.1, 13–18

A Fast Hybrid Arithmetic Unit for Elliptic Curve Cryptosystem in Galois Fields with Prime and Composite Exponents

Moon Gyung Kim,[1a)] **Su Jung Yu,**[2] **Yong Surk Lee,**[1] **and Joo Seok Song**[2]

[1] *Department of Electrical and Electronic Engineering, Yonsei University, 134 Shinchon-dong, Sudaemoon-ku, Seoul, Korea*

[2] *Department of Computer Science, Yonsei University, 134 Shinchon-dong, Sudaemoon-ku, Seoul, Korea*

a) *bungae@yonsei.ac.kr*

Abstract: In this paper, we propose the efficient flexible hybrid arithmetic unit for elliptic curve cryptographic applications which executes addition, multiplication and inversion in GF(2^m). The proposed flexible hybrid GFAU is beneficial in cost-effectively implementing microprocessors' elliptic curve public-key extensions. It targets applications for smart cards whose gate count is below 20,000.

Keywords: finite field arithmetic, elliptic curves, hybrid GFAU, multiplication, inversion, cryptography

Classification: Science and engineering for electronics

References

[1] C. Paar, P. Fleischmann, and P. Soria-Rodriguez, "Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents," *IEEE Trans. on Comp.*, vol. 48, no. 10, pp. 1025–1034, Oct. 1999.

[2] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[3] E. Mastrovito, "VLSI Architectures for Computation in Galois Fields," PhD thesis, Dept. of Electrical Eng., Linkoping Univ., Sweden, 1991.

[4] S. Moon, J. Park, and Y. Lee, "Fast VLSI Arithmetic Algorithms for High-Security Elliptic Curve Cryptographic Applications," *IEEE Trans. on Consumer Electron.*, vol. 47, pp. 700–708, Aug. 2001.

[5] SECG SEC2: Recommended Elliptic Curve Cryptography Domain Parameters, v1.0, Sept. 20, 2000.

[6] J. Goodman and A. Chandrakasan, "An Energy Efficient Reconfigurable Public-Key Cryptography Processor," *IEEE J. of Solid-State Circuit*, vol. 36, no. 11, Nov. 2001.

[7] Chi Huang, Jimmei Lai, Junyan Ren, and Qianling Zhang, "Scalable Elliptic Curve Encryption Processor for Portable Application," 5th Int. Conf. ASIC, pp. 1312–1316, Oct. 2003.

DOI: 10.1587/elex.1.13
Received January 31, 2004
Accepted February 03, 2004
Published April 10, 2004

IEICE Electronics Express, Vol.1, No.1, 13–18

1 Introduction

Elliptic Curve Cryptosystems (ECC) are becoming increasingly more popular in the limited memory and computational power applications. ECC arithmetic based on prime fields or extension fields of GF(2), denoted by GF(p) or GF(2^m)[1]. This contribution focuses on specific computer architecture for the GF(2^m) of Elliptic curve (EC) algorithms. ECC Schemes based on the assumed difficulty of the EC Discrete Logarithm Problem (ECDLP)[2]. The hardware architecture for arithmetic in Galois fields GF(2^m) provide for efficient computation speed. One of the hardware architectures is the hybrid method. The concept of hybrid architecture was represented by Mastrovito[3], and a detailed explanation was done by Paar[1]. In this paper, We proposed Hybird GFAU architecture is reinforced by modifying the implementation slightly and can work on not only composite exponents but also non-composite exponents of GF(2^m). We merge both hybrid multiplier and hybrid divider into a hybrid GFAU, and we modify the hybrid GFAU into the flexible hybrid GFAU that can execute 113 - 571 bit-long keys. We compare performance of our flexible GFAU to DSRCP [6] and SECEP [7], which can perform encryption with flexible key sizes as similar to ours.

2 The hybrid Multiplier and Divider in GF(2^m)

Finite field multipliers can be classified into three kinds : the bit-serial multipliers with $O(m)$ area requirement, the bit-parallel multipliers with $O(m^2)$ area requirement, and hybrid, which are partially bit-serial and partially bit-parallel[3]. The hybrid multiplier, which Paar[1] first announced, is twice as fast as normal GF serial multipliers, but it just works on composite exponents like GF($(2^n)^m$).

Finite field division in the GF(2^m) has the form $A(x)/B(x)$ modular $P(x)$, where the degrees of $A(x)$ and $B(x)$ are small than m, and $P(x) = x^m + \sum_{i=0}^{m-1} p_i x^i$ is an irreducible polynomial of degree m with $p_i \in$ GF(2). The efficient hardware can use for finite field inversion operation. The basic architecture of a GF divider is based on a 2-bit shift architecture[4]. During each step, the resulting values are saved to R, S, U, V and the count register can be calculated. The divider does not need an additional 1-bit shift operation, because the r_m bit is assumed to be zero in the first step[4].

3 The Proposed Fast Arithmetic Unit Using Hybrid method

3.1 The Proposed Hybrid Multiplier

we implemented not only $x^2A(x)$ operation but also $xA(x)$ operation. This implementation is done not only by combining two separate kinds of circuits and selecting one, but by implementing each circuit and sharing common blocks. Initially, a common serial multiplier works based on the Eq.(1).

$$Z_i(x) = b_{m-1-i}A(x) + xZ_{i-1}(x) \tag{1}$$

DOI: 10.1587/elex.1.13
Received January 31, 2004
Accepted February 03, 2004
Published April 10, 2004

Here, to reform a serial multiplier into a hybrid multiplier, we modified the Eq.(1) to derive the modified Eq.(2)

$$Z_i(x) = (b_{m-1-2i}x + b_{m-1-2i-1})A(x) + xZ_{i-1}(x) \tag{2}$$

This Eq.(2) is presented in completed form in Eq.(3).

$$Z_i(x) = b_{m-1-2i}\sum_{j=0}^{m-2} a_j x^{j+1} + b_{m-1-2i}a_{m-1}x^m + b_{m-1-2i-1}A(x) + \sum_{j=0}^{m-3} Z_j x^{j+2} + Z_{m-2}x^m + Z_{m-1}x^{m+1} \tag{3}$$

There is a common block in Eq.(3) which is independent from bit-location j and can be shared among bit-functional blocks. Those overlapping blocks statistically significantly reduce overall area of the circuit. This method can be applied to operations such as $x^3A(x)$ and $x^4A(x)$.
We designed a bit-functional block by using above Eq.(3). All operations except for last one are completed by using 2-bit shift operation. Last one is calculated using 1-bit shift operation to satisfy the odd-number length.

3.2 The Proposed Hybrid Divider

We implemented the new hybrid GF divider on the basis of the above standard hybrid architecture. To be certain of our hypothesis, we need to define an all-inclusive equation for U/x and U/x^2. First, we can calculate U/x considering the fact that the coefficient p_0 in $P(x)$ is always one, and obtain the following Eq.(4).

$$U(x)/x = \sum_{i=1}^{m-1} u_i x^{i-1} + u_0/x = \sum_{i=0}^{m-2} (u_{i+1} + u_0 p_{i+1})x^i + u_0 x^{m-1} \tag{4}$$

Next, we can calculate U/x^2 while dividing U/x with x, and obtain Eq. (5).

$$\begin{aligned} U(x)/x^2 &= (U(x)/x)/x \\ &= \sum_{i=1}^{m-2} (u_{i+1} + u_0 p_{i+1})x^{i-1} + u_0 x^{m-2} + (u_1 + u_0 p_1)/x \\ &= \sum_{i=0}^{m-3} (u_{i+2} + u_0 p_{i+2} + (u_1 + u_0 p_1)p_{i+1})x^i \\ &\quad +(u_0 + (u_1 + u_0 p_1)p_{m-1})x^{m-2} + (u_1 + u_0 p_1)x^{m-1} \end{aligned} \tag{5}$$

Using above equations, we implemented the RS cell and UV cell. We can determine that the RS cell does not require any modulo operations, so we simplified the cell.

3.3 The Proposed Hybrid GFAU

We merged the multiplier and the divider, which are implemented as presented above. First, the hybrid divider can calculate xU, xV, x^2U, and x^2V

DOI: 10.1587/elex.1.13
Received January 31, 2004
Accepted February 03, 2004
Published April 10, 2004

operations, and those operations are the essential operations of our hybrid multiplier. To execute hybrid multiplication within the hybrid divider, therefore we need to add control logic and modify the UV cell. The control signal in the divider is generated from $r_m r_{m-1}$, not $r_{m-1} r_{m-2}$, so the processing sequence is changed to the beginning 1-bit shift once, and continues as a 2-bit shift. The resulting modification is applied to the UV cell and the final logic of the RS cell and UV cell is shown in Fig. 1

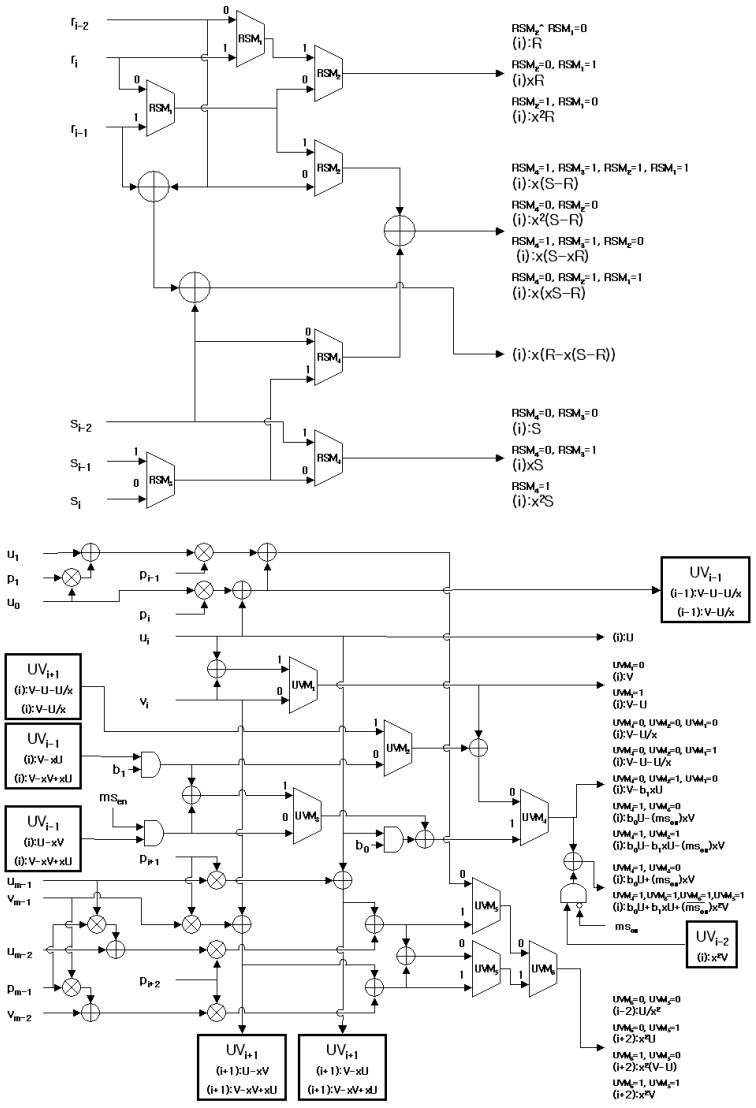

Fig. 1. The Hybrid GFAU architecture using RS cell and modified UV cell.

Basically, the implemented GFAU executes a division operation. However, it can execute addition, subtraction or multiplication if the control signal is properly applied. Addition and subtraction can be processed when operands are put into the U and V registers and passed through UV cell. Multiplication can be calculated when V register is initialized to zero and the multiplicand and multiplier values are loaded into U and R registers,

DOI: 10.1587/elex.1.13
Received January 31, 2004
Accepted February 03, 2004
Published April 10, 2004

respectively. If the V register is not initialized to zero, the multiplication process produces a multiply-and-accumulation result, which aid in reducing the overall number of calculations.

3.4 The Flexible Design of GFAU

Standard for Efficient Cryptography Group(e.q. SECG[5]) founded Certicom Corporation to solve the facing problem with which vendors or users are faced while developing interactive security solution[5].
We modify the hybrid GFAU to support those polynomials by SECG. Also, we need to shrink the GFAU small enough to locate in a smart card applications. To implement that modified GFAU, we divide operands into words which are sliced by 128-bit word length. The modified GFAU execute word-length operations. The modified GFAU is called a flexible GFAU, which can execute all word-length operations with any reduction polynomials.

4 Measurements and Comparisons

4.1 Synthesis Result

We implement all hybrid units for GF(2^{193}) and the flexible GFAU unit for 128-bit word-length with Verilog HDL, and synthesize them with Design Analyzer of Synopsys. The library used in synthesis is a Hynix 0.25um standard cell library, and the synthesis condition is tested in the worst case condition, 100C and 2.30V. Table I shows the synthesized result of the hybrid multiplier, the hybrid multiplier, the hybrid GFAU, and the flexible hybrid GFAU.

Table I. Synthesis Result of The Hybrid Units

	Hybrid Multiplier	Hybrid Divider	Hybrid GFAU	Flexible GFAU
Gate count	8,829	31,114	30,412	16,893
Critical path delay	16.41ns	15.71ns	17.44ns	9.24ns
Clock Frequency	60.9MHz	63.6MHz	57.3MHz	108.2MHz
Power consumption	5.46mW	8.88mW	14.91mW	10.58mW

4.2 Performance Comparison

We can reduce memory access by using these calculation results as input operands. If we assume that the memory access cycle is s, it takes 20s+2m+5 cycles for a point addition operation, and 13s+2.5m+2 cycles for a point double operation. The calculation process is like the hybrid GFAU, but the flexible GFAU cannot process RS cell and UV cell simultaneously because it can load only two operands at once. Moreover, the flexible GFAU divides operands into word size and loads the pieces of operands sequentially. With these results, if we assume the number of words is w, it takes m(s+1)(8w+1) cycles for a point addition operation, and m(s+1)(10w+1.5) cycles for a point double operation.

DOI: 10.1587/elex.1.13
Received January 31, 2004
Accepted February 03, 2004
Published April 10, 2004

To calculate scalar multiplication in GF(2^m) ECC, we execute $(m/2)$ point additions and $(m-1)$ point doubles on average. If we assume that the memory access cycle s is one, we can obtain $4m^2 * (7w+1) - m(20w+3)$ cycle count on average with the flexible GFAU. We sacrifice performance for the incredible reduction of area and power consumption to locate in a smart card applications. The performance comparison is shown in Table II

Table II. Performance Comparison Among GF Processors

	Hybrid GFAU	DSRCP [6]	SECEP [7]
Gate count	16,983	880,000	56,000
Peak operating frequency (MHz)	108.2	50	100
Power(mW)	10.58	74	>14
Time of one point multiplication			
(@256 bit)	39.17ms	14.5ms	
(@251 bit)	37.65ms		5.5ms

5 Conclusion

In this paper, we implement hybrid logics for GF(2^m) arithmetic and expand it being flexible. The base architecture originates from a hybrid multiplier, and its architecture is modified to implement a hybrid divider. The hybrid GFAU is based on a hybrid divider, and it can execute addition, multiplication and division. Then, we sacrifice the performance for the flexible GFAU to execute all reduction polynomials and to be small enough to locate in a smart card applications.

Acknowledgments

The authors would like to thank Hynix Semiconductor for their generous assistance and financial support.

DOI: 10.1587/elex.1.13
Received January 31, 2004
Accepted February 03, 2004
Published April 10, 2004