

Novel iterative digit-serial modular division over $\text{GF}(2^m)$

Guerric Meurice de Dormale^{*}, Jean-Jacques Quisquater

Université Catholique de Louvain, UCL Crypto Group,
Laboratoire de Microélectronique (DICE),
Place du Levant 3, B-1348 Louvain-La-Neuve, Belgium.
{gmeurice, quisquater}@dice.ucl.ac.be

Abstract. Public key cryptography is a concept used by many useful functionalities such as digital signature, encryption, key agreements, ... For those needs, elliptic curve cryptography is an attractive solution: for a given level of security, it provides one of the best key sizes and amount of exchanged data.

Cryptosystems based on elliptic curve necessitate a costly modular division. Depending on the choice of coordinates, this operation is needed at each step of algorithms or at the end of the whole computation. As a result, efficient modular division implementations could be useful for some area constrained designs and are required for high-speed processors with output in affine coordinates.

For that purpose, an improved modular division algorithm is presented. This new design is particularly well suited for hardware implementations as illustrated by our FPGA realization.

To the best of our knowledge, it is the first report about an iterative digit-serial modular division algorithm.

Keywords: Modular division, digit-serial, $\text{GF}(2^m)$, polynomial basis, arithmetic, elliptic curve cryptography, FPGA, efficient implementations.

1 Introduction

With the advent of public key cryptography, many useful functionalities appeared such as digital signature, public key encryption, key agreements, ... For those needs, elliptic curve cryptography (ECC) is an attractive solution. Co-invented by V. Miller [16] and N. Koblitz [15] in 1985, it provides one of the highest security per bit of known public key scheme. This means highly wanted properties like less processing power, storage, bandwidth and power consumption. For a thorough description of the topic, the reader is referred to [4] for mathematical background and to [12] for implementation issues.

As the underlying operations of public key cryptosystems are computer-intensive, hardware processor or co-processor are often needed to speed up the computation. As a result, efficient algorithms and specific implementations of the critical arithmetic operations are needed.

The main operation of ECC is the scalar point multiplication. The underlying modular arithmetic operations are addition, squaring, multiplication and the costly division. Depending on the choice of coordinates, a modular division is needed for each point addition and point doubling or only at the end of the scalar multiplication. As a result,

^{*} Supported by the Belgian fund for industrial and agricultural research (FRIA)

a cost-effective modular division could be useful for area constrained designs based on affine coordinates (e.g. [1]). High-speed processors using projective coordinates and handling the conversion of the final result back to affine representation (e.g. [10]) require also efficient modular division. Indeed, this operation must not constrain the area and working frequency of the processor.

The first aim of this paper is to emphasize the most efficient modular division algorithm over $\text{GF}(2^m)$. On this basis, a novel algorithm processing simultaneously two bits is presented. To the best of our knowledge, it is the first proposal of an iterative digit-serial algorithm. This method could be further exploited to carry out algorithms processing more than two bits (with an exponential complexity), but it is not the topic of this paper.

The second aim is to provide efficient implementations for the selected hardware platform. The algorithms have been optimized in order to map the resources of Xilinx Virtex FPGAs in the best possible way. Resulting designs exhibit a minimal chip area and a high working frequency. Moreover, the area increase of the 2-bit digit-serial design compared with the serial design is only 55 %.

This paper is structured as follows: Section 2 introduces the previous works on modular division over $\text{GF}(2^m)$ and emphasizes the most efficient one. Section 3 displays the selected algorithm and presents the novel digit-serial modular division. The optimized designs for the selected platform stands in Section 4. The results are discussed in Section 5 and finally, conclusions are in Section 6.

2 Previous Works

There are three main methods to perform the modular division X/Y over $\text{GF}(2^m)$:

- Little Fermat’s theorem, computed by a modular exponentiation:
 $X \cdot Y^{2^m-2} \bmod p(x)$ [14].
- Solution of a system of m linear equations over $\text{GF}(2)$ using Gaussian elimination [13].
- Iterative transformations of the *gcd* (greatest common divisor) function based on Euclid [7] or Stein [18] algorithms.

The latter exhibits the best area-time complexity [19]. So, this paper focusses on *gcd* based algorithms and especially on the binary versions.

In the literature, this kind of algorithms are usually considered as very slow (e.g. [8]). One reason is that modular divisions using the basic Euclid and Stein algorithms are not favorable to efficient implementations. They need degree comparisons at each steps, increasing the area-time complexity and thwarting digit-serial technics. Surprisingly, a method replacing this comparison by a much more simple counter has been described as early as in the mid 80’s by Brent and Kung [3].

Exploiting this counter idea, Brunner et al. [2] presented an efficient binary shift-left algorithm. Recently, Wu et al. [19]¹ and Kim et al. [6] presented two very similar serial binary shift-right algorithms. The authors show that this modification leads to

¹ Notice that, because their r and s variables converge towards zero, they should reduce in a triangular shape the size of their systolic array design.

an even better area-time complexity.

The next step was to adapt this algorithm to a digit-serial version, favorable to an iterative design. For an m -degree irreducible polynomial $p(x)$, the aim of the digit-serial technique is to divide the $2m - 1$ steps required by the algorithm by the digit size d . The complexity of the circuit quickly increase with d , but small d can already achieve an interesting improvement on the latency.

It should be emphasized here that the proposed method is different from a “multi bit-shift” technique. As showed by Gutub [11], it saves at most 15% of the $2m - 1$ steps instead of 50% with a digit size of 2 for example.

A systolic digit-serial algorithm has already been presented in [9], but the atomic piece of circuits needs more than one cycle to process the data. As a result, it is not suitable for an iterative implementation. Moreover, they do not provide an explicit algorithm: the digit-serial feature is obtained by domain partition of a systolic dependence graph. So, their optimization is not at the algorithmic level but only at the circuit level.

3 Modular Division Algorithms

Let $p(x) \in GF(2)[x]$ be an irreducible polynomial of degree m generating the field $GF(2^m)$. Let X and Y be two polynomials with coefficient in $GF(2)$ represented by $X = (x_{m-1}, \dots, x_1, x_0)$ and $Y = (y_{m-1}, \dots, y_1, y_0)$. Only the polynomial basis is considered.

Algorithm 1 is equivalent to the binary shift-right algorithm of Wu et al. and Kim et al. It is used to process the division $X/Y \bmod p(x)$ and it needs $2m - 1$ iterations to compute the result.

Algorithm 1 Serial modular division over $GF(2^m)$

Input: X, Y, p
Output: $X/Y \bmod p$
 $U \leftarrow Y, V \leftarrow p, R \leftarrow X, S \leftarrow 0, k \leftarrow 2m - 2, D \leftarrow 1, IsPos \leftarrow 0$
while $k \geq 0$ **do**
 if $u_0 = 0$ **then** $U \leftarrow U/2, R \leftarrow R/2 \pmod{p}$
 if $IsPos = 0$ **then** $D \leftarrow D + 1$
 else if $D = 0$ **then** $D \leftarrow D + 1, IsPos \leftarrow 0$ **else** $D \leftarrow D - 1$
 else if $IsPos = 1$ **then** $U \leftarrow U/2 \oplus V/2, R \leftarrow R/2 \oplus S/2 \pmod{p}$
 if $D = 0$ **then** $D \leftarrow D + 1, IsPos \leftarrow 0$ **else** $D \leftarrow D - 1$
 else $D \leftarrow D - 1, IsPos \leftarrow 1$
 $\{U \leftarrow U/2 \oplus V/2, V \leftarrow U\}, \{R \leftarrow R/2 \oplus S/2 \pmod{p}, S \leftarrow R\}$
 $k \leftarrow k - 1$
return S

The following notations are used: p stands for the irreducible polynomial $p(x)$. The addition, a bitwise *xor* operation, is written \oplus . The shift right $/2$ operator is used for the division by the polynomial root. The bit $IsPos$ is used to store the sign of the unsigned counter D . It is equal to 1 when the counter represents a positive value. Finally, the operations between brackets are performed in parallel.

Some adjustments have also been made to ease the implementations covered by the Section 4. For D , a simple unsigned arithmetic counter is used instead of a one-hot

counter. It has been chosen in order to reduce the chip area while keeping good working frequency. For the digit-serial algorithm, the variables U and R are loaded with the operands Y and X multiplied by 2.

More precisely, the signed δ counter must handle two different operations: $\delta - 1$ and $-\delta - 1$. The following operations must then be performed by the unsigned counter D , zero being considered as a positive value. To decrement the δ counter, if D represents a negative value ($IsPos = 0$), it must be incremented. If it represents a positive value but equals zero, it becomes negative ($IsPos \leftarrow 0$) and is also incremented. Otherwise, D is decremented. The negation together with the decrement of the δ counter happens only when it is negative. So, D needs only to be decremented and $IsPos$ set to express a positive value.

For the division by the polynomial root, $U/2 \equiv (0, u_{m-1}, \dots, u_1)$, the degree of the polynomials decreases by one. If the division must be performed modulo the irreducible polynomial, the degree zero of the divided polynomial must be used: $R/2 \pmod{p} \equiv (r_0, r_{m-1} \oplus r_0 p_{m-1}, \dots, r_1 \oplus r_0 p_1)$ (using the fact that coefficients of the degree m and zero of $p(x)$ always equal one).

Algorithm 2 Digit-serial modular division over $\text{GF}(2^m)$, 2-bit digit

Input: X, Y, p
Output: $X/Y \pmod{p}$
 $U \leftarrow 2Y, V \leftarrow p, R \leftarrow 2X, S \leftarrow 0, k \leftarrow m - 1, D \leftarrow 1, IsPos \leftarrow 0$
while $k \geq 0$ **do**
 if $u_1 = 0$ **then**
 if $u_2 = 0$ **then** $U \leftarrow U/4, R \leftarrow R/4 \pmod{p}$
 if $IsPos = 0$ **then** $D \leftarrow D + 1$
 else if $D = 1$ **then** $IsPos \leftarrow 0$ **else** $D \leftarrow D - 1$
 else if $IsPos = 1$ **then** $U \leftarrow U/4 \oplus V, R \leftarrow R/4 \oplus S \pmod{p}$
 if $D = 1$ **then** $IsPos \leftarrow 0$ **else** $D \leftarrow D - 1$
 else $IsPos \leftarrow 1, \{U \leftarrow U/4 \oplus V, V \leftarrow U/4\},$
 $\{R \leftarrow R/4 \oplus S \pmod{p}, S \leftarrow R/4 \pmod{p}\}$
 else if $IsPos = 1$ **then**
 if $u_2 \oplus v_1 = 0$ **then** $U \leftarrow U/4 \oplus V/2, R \leftarrow R/4 \oplus S/2 \pmod{p}$
 else $U \leftarrow U/4 \oplus V/2 \oplus V, R \leftarrow R/4 \oplus S/2 \oplus S \pmod{p}$
 if $D = 1$ **then** $IsPos \leftarrow 0$ **else** $D \leftarrow D - 1$
 else
 if $u_2 \oplus v_1 = 0$ **then** $\{U \leftarrow U/4 \oplus V/2, V \leftarrow U/2\},$
 $\{R \leftarrow R/4 \oplus S/2 \pmod{p}, S \leftarrow R/2 \pmod{p}\}$
 else $\{U \leftarrow U/4 \oplus V/2 \oplus U/2, V \leftarrow U/2\},$
 $\{R \leftarrow R/4 \oplus S/2 \oplus R/2 \pmod{p}, S \leftarrow R/2 \pmod{p}\}$
 if $D \neq 1$ **then** $D \leftarrow D - 1, IsPos \leftarrow 1$
 $k \leftarrow k - 1$
return S

The Algorithm 2 is the proposed digit-serial division with a digit size of 2. To improve readability, this algorithm is renamed as “2-bit digit”. The notations used for the Algorithm 1 are kept and the same choices for the loading and the counter are applied. The double shift right $/4$ operator is added and stands for the division by the square of the polynomial root.

In the building process of the 2-bit digit algorithm, two successive steps of the serial algorithm are concatenated. As the body of the serial algorithm is split in three different cases, it could be expected the 2-bit digit will need nine cases. Fortunately, two cases never happen and the complexity is reduced.

As two steps are processed at each iteration, the D' counter would never be incremented or decremented by 1 but only by 2. So, it has been modified to $D = D'/2$ to only handle increment and decrement of 1. Moreover, the counter D' is initialized with an odd value, so D never reaches zero. It is important to take care of the counter to ensure an optimal implementation: it is an essential element of the control logic and its performances will have a non negligible impact on the working frequency.

4 Hardware Implementations

This Section presents the hardware implementations of the two algorithms from Section 3 and shows their suitability for hardware realization. In order to exhibit the best performances, the designs have been optimized for the selected hardware platform: the Virtex family of Xilinx FPGA.

The three extension degree covered are: 163, 193 and 233. For each degree, two different kinds of irreducible polynomials are handled. First, the polynomials recommended by the NIST [17] and the SECG [5] are handled: $p(x) = x^{163} + x^7 + x^6 + x^3 + 1$, $p(x) = x^{193} + x^{15} + 1$ and $p(x) = x^{233} + x^{74} + 1$. They are hardwired in the circuit. Secondly, arbitrary polynomials are considered. Nevertheless, it has been decided that only 10% of their monomials, located in the lowest degrees, may have a variable coefficient. The other monomials are hardwired to zero. For applications, another choice for the kind of irreducible polynomial is very unlikely because random irreducible polynomials thwart many improvements (as illustrated in [10]).

For the operative circuits of the two implementations, the register-to-register combinatorial logic needs few lookup tables (LUTs) connected serially. As a result, the control circuits will dominate the critical path. More precisely, the D counter and its different operations are the most time consuming elements of the control part. To guarantee a high working frequency for each design, the D counter has been optimized with precomputation of the different test flags and commands.

4.1 Serial Modular Division

The implementation of Alg. 1 is illustrated in Fig. 1. In order to consume as few resources as possible, the design take into account the basic structure of the FPGA. Therefore, when it is possible, each register storing the bits of U, V, R and S variables are driven by a 4-input combinatorial circuit.

The main optimization is the splitting in two phases of the loading step. To save one input in the logic driving the U and R registers, the operands Y and X are loaded in the registers V and S . Then, using the available wires of the circuit, the content of V and S is moved to the appropriate U and R registers.

For hardwired irreducible polynomials, each bits of the U, V, R and S variables are dynamically specified by four inputs. With m -degree polynomials, this circuit will consume only $2m$ slices. For arbitrary polynomials, 10% of m additional slices are required to handle the low-degree monomials of $p(x)$.

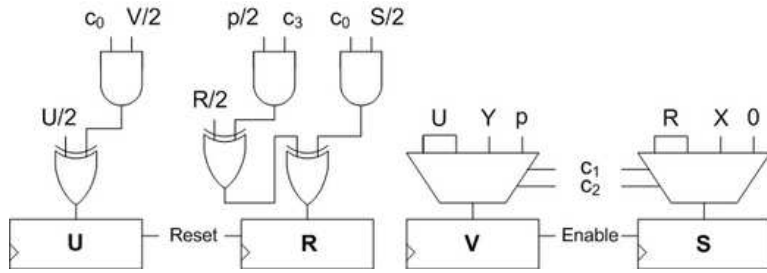


Fig. 1. Serial modular divider

4.2 Digit-Serial Modular Division

The implementation presented in Fig. 2 is based on the proposed digit-serial algorithm (Alg. 2). The same design principles as the serial algorithm are followed and another feature of the FPGA is exploited: the *xor* gate of the dedicated carry chain is used in order to extend the number of available inputs. As a result, the slices are equivalent to a combinatorial function with $2 * 4 + 1$ inputs, and 2 outputs. Indeed, if the carry input is embezzled², it is shared for the whole slice. The design is then constrained to use the two *xor* gates and to use them with the same input (otherwise half of the slice is wasted).

For hardwired irreducible polynomials, this circuit will consume only $3m$ slices plus a few additional slices for the gates driven by c_8 and c_9 (due to the sparse $p(x)$). For arbitrary polynomials, 10% of m additional slices are required for the logic driving the V and S registers. Another 10% of m additional slices are needed for the R register.

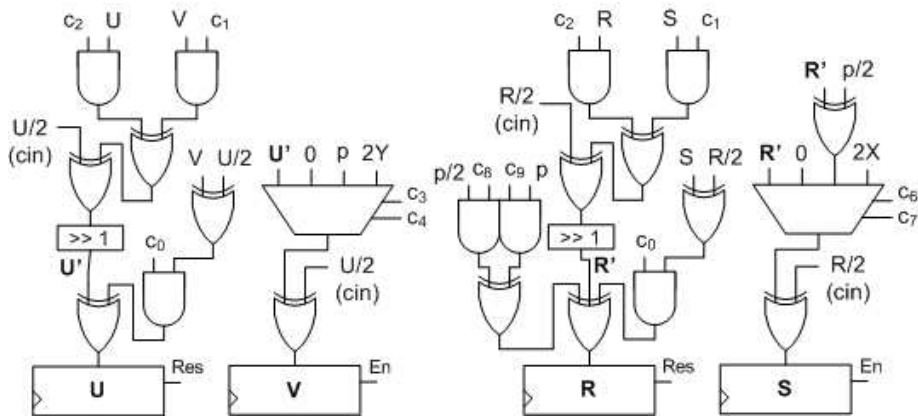


Fig. 2. Digit-serial modular divider with a digit size of 2

5 Results

The designs of the serial modular divider and the 2-bit digit modular divider have been implemented on a last generation Xilinx Virtex4 XC4VLX15-12sf363 and a small

² To achieve this optimization, constraints have to be added for the mapping tool.

and low-cost Xilinx Spartan3³ XC3S200-4ft256. The VHDL synthesis and place & route have been achieved on Xilinx ISE 7.1.03i with the best speed effort. In order to evaluate the chip area and the working frequency of the two modular divider cores, shift registers have been added for the inputs and the outputs. These resources are not considered for the area count.

The performances of each divider together with the two kinds of irreducible polynomials $p(x)$ are reported in table 1. The throughput is computed with a latency of $2m$ and $m + 1$ cycles (considering the 2 cycles needed by the load operation). For the throughput enhancement, the serial and the 2-bit digit implementations are compared for each extension field degree and $p(x)$. The 4-input LUTs count is also reported because the mapping tool of ISE gives priority to combine logic that is related. As a result, when the constraint on the working frequency is strong, many slices of the digit-serial design are half empty (but are still available for other purposes).

Table 1. Implementation results on Virtex4 (left) and Spartan3 (right)

| Designs | irreducible polynomial | Freq. Mhz | Area Slices | Area LUTs | Throughput Mbit/s | Throughput enhancement |
|-------------|------------------------|----------------|-------------|-----------|-------------------|------------------------|
| Serial | 163 - NIST | 480 210 | 504 552 | 716 712 | 240 105 | 35% 39% |
| 2-bit digit | | 325 177 | 699 648 | 1124 1093 | 323 146 | |
| Serial | 163 - arbit. | 451 193 | 543 562 | 747 742 | 225 96 | 44% 42 % |
| 2-bit digit | | 326 137 | 695 675 | 1154 1144 | 324 136 | |
| Serial | 193 - SECG | 457 178 | 641 622 | 837 832 | 228 89 | 41% 57% |
| 2-bit digit | | 324 141 | 858 751 | 1298 1277 | 322 140 | |
| Serial | 193 - arbit. | 446 167 | 675 629 | 873 869 | 223 83 | 44% 60% |
| 2-bit digit | | 324 133 | 910 798 | 1359 1337 | 322 133 | |
| Serial | 233 - NIST | 446 173 | 753 789 | 999 998 | 223 86 | 44% 61% |
| 2-bit digit | | 324 140 | 982 877 | 1526 1520 | 322 139 | |
| Serial | 233 - arbit. | 436 167 | 798 791 | 1043 1043 | 218 83 | 38% 59% |
| 2-bit digit | | 322 133 | 1055 950 | 1648 1597 | 321 132 | |

As expected, the working frequency of the digit-serial divider decreases compared with the serial divider. Nevertheless, the throughput enhancement shows that the improved latency overtakes this diminution. For each design, the reported area count follows the estimation of Section 4. Due to the best speed effort constraint, the area is slightly augmented by the logic replication of the control circuit. An important point is that the 2-bit digit designs appear to have a reasonable increase of complexity: only 55 % of additional LUTs are required. This is far from the 100 % increase expected with the simple concatenation of two serial circuits.

Comparing the performances between implementations based on the recommended irreducible polynomials (NIST, SECG) and the chosen arbitrary polynomials, the differences are no significative. This follows also the predictions.

The only found FPGA implementation of an equivalent serial modular divider is presented in [10]. While their performances are much less interesting, the comparison

³ Spartan3 devices have the same structure as the Virtex family.

in not really fair: they use completely arbitrary polynomials and they do not try to exceed a working frequency of 66 Mhz.

6 Conclusion

An iterative digit-serial algorithm for the modular division over $GF(2^m)$ has been proposed. The presented 2-bit digit version reduces the latency by a factor 2 and is well suited for hardware implementations. Among other field of applications, this algorithm is particularly interesting for elliptic curve cryptography and more precisely for some area constrained designs and high-speed processors with output in affine coordinates.

To show the efficiency of the novel algorithm, it has been implemented on FPGA of the Virtex family, with different kinds of irreducible polynomials and extension field degrees. The designs have been optimized for this platform and resulting performances exhibit a high working frequency and small area requirements.

The 2-bit digit implementation appears to have a throughput improvement between 35 % and 60 % compared with the serial implementation while only 55 % of additional area is required. It can thus be used when a higher throughput is required, with a better area efficiency.

References

1. H. Aigner, H. Bock, M. Hütter, J. Wolkerstorfer, *A Low-Cost ECC Coprocessor for Smartcards*, Cryptographic Hardware and Embedded Systems - CHES 2004, LNCS 3156, pp. 107-118, 2004.
2. H. Brunner, A. Curiger, M. Hofstetter, *On Computing Multiplicative Inverses in $GF(2^m)$* , IEEE Transactions on Computers, vol. 42, no. 8, pp. 1010-1015, 1993.
3. R.P. Brent and H.T. Kung, *Systolic VLSI Arrays for Polynomial GCD Computation*, IEEE Transactions on Computers, vol. 33, no. 8, pp. 731-736, 1984.
4. I.F. Blake, G. Seroussi, N.P. Smart, *Elliptic Curves in Cryptography*, London Mathematical Society, Lecture Notes Series 265, Cambridge University Press, 1999.
5. Certicom Research, *SEC 2: Recommended Elliptic Curve Domain Parameters*, v1.0, 2000.
6. C.H. Kim, S. Kwon, J.J. Kim, C.P. Hong, *A Compact and Fast Division Architecture for a Finite Field $GF(2^m)$* , ICCSA 2003, LNCS 2667, pp. 855-864, 2003.
7. D.E. Knuth, *The Art of Computer Programming*, vol. 2 Reading, Mass.: Addison-Wesley, 2nd edition, 1981.
8. K. Fong, D. Hankerson, J. López, A. Menezes, *Field Inversion and Point Halving Revisited*, IEEE Transactions on Computers, vol. 53, no. 8, pp. 1047-1059, 2004.
9. J.-H. Guo and C.-L. Wang, *Novel digit-serial systolic array implementation of Euclid's algorithm for division in $GF(2^m)$* , IEEE International Symposium on Circuits and Systems - ISCAS 1998, pp. 478-481, 1998.
10. N. Gura, S. C. Shantz, H. Eberle, S. Gupta, V. Gupta, D. Finchelstein, E. Goupy, D. Stebila *An End-to-End Systems Approach to Elliptic Curve Cryptography*, Cryptographic Hardware and Embedded Systems - CHES 2002, LNCS 2523, pp. 349-365, 2002.
11. A. A.-A. Gutub, *New Hardware Algorithms and Designs for Montgomery Modular Inverse Computation in Galois Fields $GF(p)$ and $GF(2^n)$* , Ph.D. Thesis, Oregon State University 2002.
12. D. Hankerson, A. Menezes, S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer Professional computing, Springer, 2004.
13. M.A. Hasan and V.K. Bhargava, *Bit-Serial Systolic Divider and Multiplier for Finite Fields $GF(2^m)$* , IEEE Transaction on Computers, vol. 41, no. 8, pp. 972-980, 1992.
14. T. Itoh and S. Tsujii, *A Fast Algorithm for Computing Multiplicative Inverses in $GF(2^m)$ Using Normal Bases*, Information and Computation, vol. 78, pp. 171-177, 1988.
15. N. Koblitz, *Elliptic curve cryptosystems*, Math. of Computation, vol. 48, pp. 203-209, 1987.
16. V. Miller, *Uses of elliptic curves in cryptography*, Advances in Cryptology - CRYPTO '85, LNCS 218, pp. 417-426, 1986.
17. U.S. Department of Commerce/National Institute of Standards and Technology (NIST), *Digital Signature Standard (DSS)*, FIPS PUB 182-2change1, 2000.
18. J. Stein, *Computational problems associated with Racah algebra*, J. Computational Physics, vol. 1, pp. 397-405, 1967.
19. C.-H. Wu, C.-M. Wu, M.-D. Shieh, Y.-T. Hwang, *High-Speed, Low-Complexity Systolic Designs of Novel Iterative Division Algorithms in $GF(2^m)$* , IEEE Transactions on Computers, vol. 53, no. 3, pp. 375-380, 2004.