High-speed Hardware Implementation of Rainbow Signature on FPGAs

Shaohua Tang[1], Haibo Yi[1], Huan Chen[1], Guomin Chen[1], and Jintai Ding[2,3]

[1] School of Computer Science & Engineering,
South China University of Technology, Guangzhou, China
shtang@IEEE.org, haibo.yi87@gmail.com, huangege@qq.com,
sarlmolapple@gmail.com
[2] Department of Applied Mathematics,
South China University of Technology, Guangzhou, China
[3] Department of Mathematical Sciences,
University of Cincinnati, OH, USA
jintai.ding@mail.uc.edu

Abstract. We propose a new efficient hardware implementation of Rainbow signature scheme. We enhance the implementation in three directions. First, we develop a new parallel hardware design for the Gaussian elimination to solve a $n \times n$ linear system with only n clock cycles. Second, a novel multiplier was designed to speed up multiplication of three elements over a finite field. Third, we further optimize the parallelization process of the hardware to generate the Rainbow signature. By integrating these optimizations, we build a new hardware implementation, which takes only 198 clock cycles to generate a Rainbow signature, a new record in generating digital signatures and four times faster than the 804- clock-cycle Balasubramanian-Bogdanov-Carter-Ding-Rupp design with similar parameters.

Key words: Multivariate Public Key Cryptosystems (MPKCs), Field-Programmable Gate Array (FPGA), digital signature, Rainbow, finite field, Gaussian elimination.

1 Introduction

Due to the fast growth of broad application of cryptography, the concern for secure and efficient hardware architectures and implementations of cryptosystems becomes increasingly critical for both symmetric and asymmetric cryptosystems, for example, DES [1], [2], AES [3], [4], RSA [5], [6], and ECC [7], [8].

In terms of asymmetric cryptosystems, most schemes currently used are based on the hardness of factoring large numbers or discrete logarithm problems . However, a potential powerful quantum computer could put much of currently used public key cryptosystems in jeopardy due to the algorithm by Peter Shor.

Multivariate Public Key Cryptosystems (MPKCs) [9] is one of main families of public key cryptosytsems that have the potential to resist the attacks by

quantum computation. They are based on the difficulty of the problem of solving multivariate quadratic equations over finite fields, which is in general NP-hard.

The hardware implementation of multivariate signature schemes is the focus of this paper. Rainbow signature scheme [9] is one of the Oil-Vinegar family signature schemes. The Oil-Vinegar family of Multivariate Public Key Cryptosystems consists of three families: balanced Oil-Vinegar, unbalanced Oil-Vinegar and Rainbow, a multilayer construction using unbalanced Oil-Vinegar at each layer. There have been some previous work to implement multivariate signature schemes, e.g. TTS [10], [11], some instances of MPKCs [12], SSE implementation of multivariate PKCs on modern x86 CPUs [13], and Rainbow scheme [14].

Since multiplication, multiplicative inverse and solving linear system of equations over finite fields are playing the key role in generating multivariate signature scheme, they are the focus in this paper. Much has been done in research on finite field multiplication [15], [16], [17], [18], [19], finite field multiplicative inverse [20], [21], [22], [23], [24], and solving linear system of equations over finite fields [25], [26], [27], [28], [29]. To achieve higher speed of these basic operations, we push further the optimizations of these key operations.

We enhance the implementation in three directions. First, we develop a new parallel hardware design for the Gaussian elimination to solve a $n \times n$ linear system with only n clock cycles. Second, a novel multiplier was designed to speed up multiplication of three elements over a finite field. Third, we further optimize the parallelization process of the hardware to generate the Rainbow signature. By integrating these optimizations, we build a new hardware implementation, which takes only 198 clock cycles to generate a Rainbow signature, a new record in generating digital signatures and four times faster than the 804- clock-cycle Balasubramanian-Bogdanov-Carter-Ding-Rupp design [14] with similar parameters. In addition, our implementation achieves an average-size utilization area.

The rest of this paper is organized as follows: In Section 2, we present the overview of Rainbow signature scheme and the basic notations used in this paper. In Section 3, the proposed hardware design for Rainbow signature scheme is presented. In Section 4, we develop an implementation design in a low-cost Field-Programmable Gate Array (FPGA) and experimental results are presented. In Section 5, the implementation is evaluated and compared with other hardware implementations. In Section 6, conclusions are summarized.

2 Overview of Rainbow Signature Scheme

2.1 Basic Notations

In order to maintain consistency, the following basic notations are used this paper.

The finite field, $GF(2^8)$, including its additive and multiplicative structure, is denoted by k.

The number of variables used in the signature construction, which is also equal to the signature size, is denoted by n.

For a Rainbow scheme, the number of Vinegar variables used in the ith layer of signature construction is denoted by v_i; the number of Oil variables used in the ith layer of signature construction is denoted by o_i, and $o_i = v_{i+1} - v_i$; the number of layers is denoted by u, a message (or the hash value of a meesage) is denoted by Y; the signature of Rainbow is denoted by X'; O_i is a set of Oil variables in the the ith layer; S_i is a set of Vinegar variables in the the ith layer.

2.2 Overview of Rainbow Scheme

The Rainbow scheme belongs to the class of oil-vinegar signature constructions. The scheme consists of a quadratic system of equations involving Oil and Vinegar variables that are solved iteratively. The Oil-Vinegar polynomial can be represented by the form

$$\sum_{i \in O_l, j \in S_l} \alpha_{ij} x_i x_j + \sum_{i,j \in S_l} \beta_{ij} x_i x_j + \sum_{i \in S_{l+1}} \gamma_i x_i + \eta.$$

The Rainbow scheme consists of four components: private key, public key, signature generation and signature verification.

Private Key The private key consists of the maps ${L_1}^{-1}$, ${L_2}^{-1}$ and F, which is held by the signer. L_1: $k^{n-v_1} \rightarrow k^{n-v_1}$ and L_2: $k^n \rightarrow k^n$ are two randomly chosen invertible affine linear maps. F is a map consists of $n - v_1$ Oil-Vinegar polynomials. F has $u - 1$ layers of Oil-Vinegar construction. The first layer consists of o_1 polynomials where $\{x_i | i \in O_1\}$ are the Oil variables, and $\{x_j | j \in S_1\}$ are the Vinegar variables. The lth layer consists of o_l polynomials where $\{x_i | i \in O_l\}$ are the Oil variables, and $\{x_j | j \in S_l\}$ are the Vinegar variables.

Public Key The public key consists of the field k and the $n - v_1$ polynomial components of $\bar{F}$, where $\bar{F} = L_1 \circ F \circ L_2$.

Signature Generation The message is defined by $Y = (y_1, ..., y_{n-v_1}) \in k^{n-v_1}$. The signature is a solution to the equation $\bar{F} = L_1 \circ F \circ L_2 = Y$.

In order to solve the equation, we should firstly compute $\bar{Y'} = {L_1}^{-1}(Y)$.

Next, to solve the equation $\bar{Y'} = F$. At each layer, the v_i Vinegar variables in the Oil-Vinegar polynomials are randomly chosen. Then, the variables at upper layer are chosen as part of the Vinegar variables. After that, the polynomials are substituted into a set of linear equations of Oil variables. If these equations have no solution, a new set of Vinegar variables should be chosen. This procedure for each successive layer is repeated until the solutions of all the equations are found. In this step, the values $\bar{X} = (\bar{x_1}, ..., \bar{x_n})$ of all the variables are found.

Finally, to compute $X' = {L_2}^{-1}(\bar{X}) = ({x_1}', ..., {x_n}')$. Then X' is the signature for document Y.

Signature Verification To verify the authenticity of a signature X', $\bar{F}(X') = Y'$ is computed. If $Y' = Y$ holds, the signature is accepted, otherwise rejected.

3 Proposed Hardware Design for Rainbow Signature

3.1 Overview of the Hardware Design

The aim of our hardware design is to further speed up the execution of the Rainbow signature. Since the multiplication, multiplicative inverse, and Gaussian elimination over finite fields are basic operations in generating multivariate signature, these operations are optimized and parallelized. In addition, a lot of parallel multipliers are designed to efficiently implement ${L_1}^{-1}$, ${L_2}^{-1}$ and F mappings. Finally, the parallelization process of the hardware to generate the Rainbow signature is further optimized.

The flowchart to generate the Rainbow signature is illustrated in Fig. 1. On the whole, the hardware execution is controlled by finite state machines.

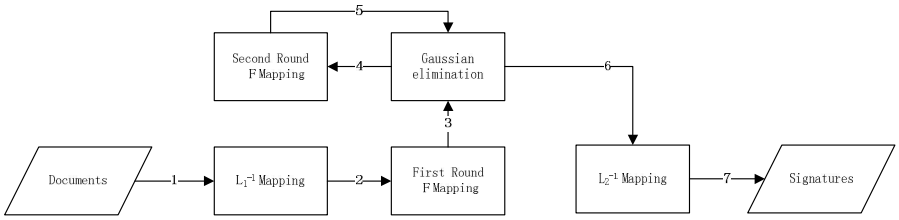

Fig. 1. The Flowchart to Generate Rainbow Signature

3.2 Parameters of Rainbow Signature

Table 1. Parameters of Rainbow in Proposed Hardware Design

Parameter	Rainbow
Ground field size	$GF(2^8)$
Message size	24 bytes
Signature size	42 bytes
Number of layers	2
Set of variables in each layer	(17, 12), (1, 12)

We adopt the parameters of the Rainbow signature in [30] to design our hardware. It is a two-layer scheme, and owns a security level above 2^{80}. There

are 17 random-chosen Vinegar variables and 12 Oil variables in the first layer, and 1 random-chosen Vinegar variables and 12 Oil variables in the second layer. The parameters are shown in Table 1.

3.3 Choice of Primitive Polynomial for the Ground Field

The choice of the primitive polynomial for the ground field k is a critical part of our hardware design, since it determines the structure of the finite field and affects the efficiency of the operations over the ground field. Our principle is to select the primitive polynomial with fewest number of coefficients valued one, namely a polynomial with lowest weight.

The primitive polynomials over $GF(2^8)$ can be expressed as the form $x^8+x^k+...+1$, where $0 < k < 8$. They can also be expressed as 9-bit binary digits, where the first bit and the last bit are valued one. There are totally 16 candidates. By comparing the efficiency of signature generations basing on different primitive polynomials, $x^8 + x^6 + x^3 + x^2 + 1$ is finally chosen as the primitive polynomial in our hardware design.

3.4 Design of Multiplication over Finite Fields

The multiplication over the finite field is the very basic operation in multivariate signature. An optimized design of the multiplier can dramatically improve the overall hardware execution efficiency.

Suppose $a(x)$ and $b(x)$ are two elements in $GF(2^8) = GF(2)[x]/f(x)$, and $c(x) = a(x) \times b(x) \bmod f(x)$ is the expected multiplication result, where $f(x)$ is the primitive polynomial. In the case of the chosen primitive polynomial $f(x) = x^8 + x^6 + x^3 + x^2 + 1$, the multiplication is computed as follows;

We first compute

$s_0 = a_0b_0$,
$s_1 = a_1b_0 \oplus a_0b_1$,
$s_2 = a_2b_0 \oplus a_1b_1 \oplus a_0b_2$,
$s_3 = a_3b_0 \oplus a_2b_1 \oplus a_1b_2 \oplus a_0b_3$,
$s_4 = a_4b_0 \oplus a_3b_1 \oplus a_2b_2 \oplus a_1b_3 \oplus a_0b_4$,
$s_5 = a_5b_0 \oplus a_4b_1 \oplus a_3b_2 \oplus a_2b_3 \oplus a_1b_4 \oplus a_0b_5$,
$s_6 = a_6b_0 \oplus a_5b_1 \oplus a_4b_2 \oplus a_3b_3 \oplus a_2b_4 \oplus a_1b_5 \oplus a_0b_6$,
$s_7 = a_7b_0 \oplus a_6b_1 \oplus a_5b_2 \oplus a_4b_3 \oplus a_3b_4 \oplus a_2b_5 \oplus a_1b_6 \oplus a_0b_7$,
$s_8 = a_7b_1 \oplus a_6b_2 \oplus a_5b_3 \oplus a_4b_4 \oplus a_3b_5 \oplus a_2b_6 \oplus a_1b_7$,
$s_9 = a_7b_2 \oplus a_6b_3 \oplus a_5b_4 \oplus a_4b_5 \oplus a_3b_6 \oplus a_2b_7$,
$s_{10} = a_7b_3 \oplus a_6b_4 \oplus a_5b_5 \oplus a_4b_6 \oplus a_3b_7$,
$s_{11} = a_7b_4 \oplus a_6b_5 \oplus a_5b_6 \oplus a_4b_7$,
$s_{12} = a_7b_5 \oplus a_6b_6 \oplus a_5b_7$,
$s_{13} = a_7b_6 \oplus a_6b_7$,
$s_{14} = a_7b_7$.

Next, we compute

$$
\begin{aligned}
c_7 &= s_7 \oplus s_9 \oplus s_{11} \oplus s_{12},\\
c_6 &= s_6 \oplus s_8 \oplus s_{10} \oplus s_{11},\\
c_5 &= s_5 \oplus s_{10} \oplus s_{11} \oplus s_{12} \oplus s_{14},\\
c_4 &= s_4 \oplus s_9 \oplus s_{10} \oplus s_{11} \oplus s_{14} \oplus s_{14},\\
c_3 &= s_3 \oplus s_8 \oplus s_9 \oplus s_{10} \oplus s_{12} \oplus s_{13} \oplus s_{14},\\
c_2 &= s_2 \oplus s_8 \oplus s_{13} \oplus s_{14},\\
c_1 &= s_1 \oplus s_9 \oplus s_{11} \oplus s_{13} \oplus s_{14},\\
c_0 &= s_0 \oplus s_8 \oplus s_{10} \oplus s_{12} \oplus s_{13}.
\end{aligned}
$$

Then $c(x) = c_7x^7 + c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$ is the multiplication result.

3.5 Design of Multiplicative Inverse over Finite Fields

The multiplicative inverse over the finite field is a time-consuming and crucial operation in multivariate signature. An optimized design of the multiplier can really help to improve the overall performance. We design the multiplicative inverse based on the Fermat's theorem [31].

Suppose $f(x)$ is the primitive polynomial and β is an element over $GF(2^8)$, where $\beta = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$. According to the Fermat's theorem, we have $\beta^{2^8} = \beta$, and $\beta^{-1} = \beta^{2^8-2} = \beta^{254}$.

Since $2^8 - 2 = 2 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7$, then

$\beta^{-1} = \beta^2\beta^4\beta^8\beta^{16}\beta^{32}\beta^{64}\beta^{128}$.

Therefore, the computation of β^{-1} can be converted to the multiplication of $\beta^2, \beta^4, \beta^8, \beta^{16}, \beta^{32}, \beta^{64}$ and β^{128}. We can then construct the logic expressions of these items. Since

$$
\begin{aligned}
\beta^{2^n} = {} & a_7x^{2^n\times 7} + a_6x^{2^n\times 6} + a_5x^{2^n\times 5} + a_4x^{2^n\times 4} +\\
& a_3x^{2^n\times 3} + a_2x^{2^n\times 2} + a_1x^{2^n\times 1} + a_0x^{2^n\times 0},
\end{aligned}
$$

we have

$$
\begin{aligned}
\beta^2 &= a_7x^{14} + a_6x^{12} + a_5x^{10} + a_4x^8 + a_3x^6 + a_2x^4 + a_1x^2 + a_0x^0,\\
\beta^4 &= a_7x^{28} + a_6x^{24} + a_5x^{20} + a_4x^{16} + a_3x^{12} + a_2x^8 + a_1x^4 + a_0x^0,\\
\beta^8 &= a_7x^{56} + a_6x^{48} + a_5x^{40} + a_4x^{32} + a_3x^{24} + a_2x^{16} + a_1x^8 + a_0x^0,\\
\beta^{16} &= a_7x^{112} + a_6x^{96} + a_5x^{80} + a_4x^{64} + a_3x^{48} + a_2x^{32} + a_1x^{16} + a_0x^0,\\
\beta^{32} &= a_7x^{224} + a_6x^{192} + a_5x^{160} + a_4x^{128} + a_3x^{96} + a_2x^{64} + a_1x^{32} + a_0x^0,\\
\beta^{64} &= a_7x^{448} + a_6x^{384} + a_5x^{320} + a_4x^{256} + a_3x^{192} + a_2x^{128} + a_1x^{64} + a_0x^0,
\end{aligned}
$$

and

$$\beta^{128} = a_7x^{896} + a_6x^{768} + a_5x^{640} + a_4x^{512} + a_3x^{384} + a_2x^{256} + a_1x^{128} + a_0x^0.$$

The computation of $x^{2^n\times i}$ over finite field should be modulo the primitive polynomial, then β^{2^n} is transformed into the equivalent form. For instance,

$$\beta^{2^n} = a'_7x^7 + a'_6x^6 + a'_5x^5 + a'_4x^4 + a'_3x^3 + a'_2x^2 + a'_1x^1 + a'_0.$$

Finally, the multiplicative inverse over $GF(2^8)$ is computed via the the multiplication of 7 elements, i.e., $\beta^2, \beta^4, \beta^8, \beta^{16}, \beta^{32}, \beta^{64}$ and β^{128}.

3.6 Optimized Gaussian Elimination

Solving system of linear equations over finite fields is a key component in multivariate signature. An optimized design of the Gaussian elimination can surely improve the efficiency of solving system of linear equations. The Gaussian elimination consists of the pivoting, normalizing, and eliminating operations. The parallelization of these operations can enhance the overall performance.

Pivoting Operation The pivoting operation is to find a nonzero element in the pivot column if the current element is zero, and then the corresponding operation is to exchange the current row with another row. The normal design according to the original principle is to exchange the rows after the previous computation results are written to the matrix storage, which is time-consuming. However, we optimize the procedure to exchange the rows before updating the matrix storage.

Normalizing Operation The normalizing operation consists of multiplicative inverse and multiplications. Then we can enhance the implementation via two aspects.

First, the multiplicative inverse β^{-1} over $GF(2^8)$ is optimized to the multiplication of 7 elements due to $\beta^{-1} = \beta^2\beta^4\beta^8\beta^{16}\beta^{32}\beta^{64}\beta^{128}$, as mentioned in the previous sub-section.

Second, a new multiplier is designed to speed up the multiplication with three elements that denoted by $ThreeMult(v1, v2, v3)$, while the multiplication with two elements is defined by $TwoMult(v1, v2)$. The schematic diagram of normalization is shown on the left-hand side of Fig. 2, where I_i stands for β^i , R_i for the ith element in the pivot row, and N_i for the normalizing result, respectively. Then, we have the expressions

$S_1 = ThreeMult(I_2, I_4, I_8)$,
$S_2 = ThreeMult(I_{16}, I_{32}, I_{64})$,
$S_4 = TwoMult(I_{128}, R_i)$,
and
$N_i = ThreeMult(S_1, S_2, S_4)$.

Suppose $inver$ is the multiplicative inverse of the pivot element, then $N_i = inver \times R_i$. Since S_1, S_2 and S_4 can be implemented parallelly in hardware, the critical path of normalizing consists of only two multiplications with three elements.

Eliminating Operation Suppose that C_i is the ith element in the pivot column, and N_j is the jth element in the pivot row after being normalized. a_{ij} is an element in the matrix A, and E_{ij} is the corresponding element after elimination.

We design a parallel way to perform the normalizing and eliminating operations simultaneously. The schematic diagram is shown in Fig. 2. Then, we have the expressions

$S_1 = ThreeMult(I_2, I_4, I_8)$,
$S_2 = ThreeMult(I_{16}, I_{32}, I_{64})$,
$S_3 = ThreeMult(I_{128}, R_j, C_i)$,
$E_{ij} = ThreeMult(S_1, S_2, S_3)$,
and
$E_{ij} = E_{ij} + a_{ij}$.

Since S_1, S_2 and S_3 can be computed parallelly in hardware, the critical path of eliminating consists of only two multiplications with three elements.

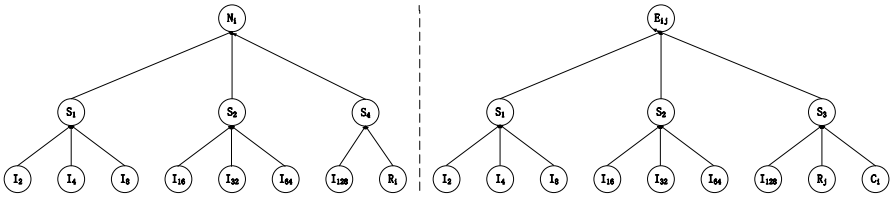

Fig. 2. Optimized Normalization and Elimination

Parallelization of Gaussian Elimination The normalizing and eliminating operations are designed to execute parallelly. The right-hand side of Fig. 3 illustrates the schematic diagram of our optimized design, and the left-hand side of Fig. 3 is the original principle of Gaussian elimination. Fig. 3 shows that the critical path of our design is reduced from five multiplications to two multiplications, where our multiplier is also optimized to compute with three elements.

Fig. 4 illustrates an example to solve a 3×3 system of linear equations in a parallel way with only 3 clock cycles. Since there are no dataflow dependencies between the normalizing and eliminating operations, they can be executed parallelly within the same clock cycle. One normalizing and two eliminating operations are executed simultaneously within each clock cycle, then it takes 3 clock cycles to conduct 3 parallel normalizing operations and 6 parallel eliminating operations. As a result, we can obtain the solution to the system of linear equations in 3 clock cycles.

Therefore, our design takes only n clock cycles to solve a $n \times n$ system of linear equations, and the time complexity is $O(n)$.

3.7 Design of L_1^{-1}, L_2^{-1} and F Mappings

$L_1{}^{-1}$, $L_2{}^{-1}$ and F mappings constitutes the private key of Rainbow signature. L_1: $k^{24} \to k^{24}$ and L_2: $k^{42} \to k^{42}$ are two randomly chosen invertible affine linear maps. The transformations of $L_1{}^{-1}$ and $L_2{}^{-1}$ invoke the matrix multiplication and offset addition. Table 2 summarizes the numbers of multiplications

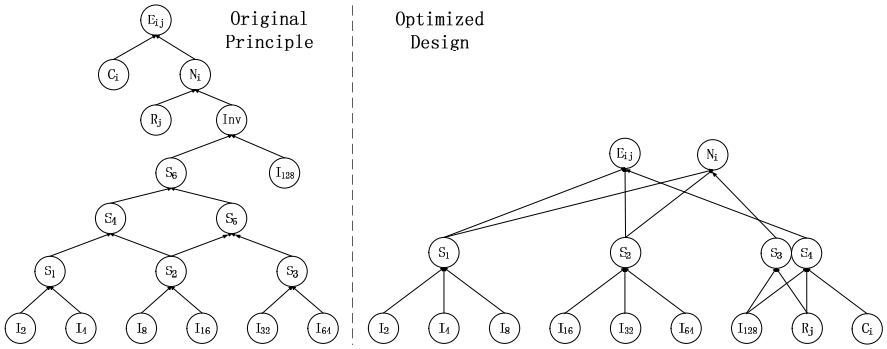

Fig. 3. Comparison of Gaussian Elimination Designs

$$\begin{pmatrix} **** \\ **** \\ **** \end{pmatrix} \rightarrow \begin{pmatrix} 1*'*'*' \\ 0*'*'*' \\ 0*'*'*' \end{pmatrix} \rightarrow \begin{pmatrix} 10*''*'' \\ 01*''*'' \\ 00*''*'' \end{pmatrix} \rightarrow \begin{pmatrix} 100*''' \\ 010*''' \\ 001*''' \end{pmatrix}$$

Fig. 4. An Example of Optimized Gaussian Elimination with $n = 3$

Table 2. Number of Multiplications in ${L_1}^{-1}$, ${L_2}^{-1}$ and F Mappings

Components	Number of multiplications
${L_1}^{-1}$ mapping	576
The first F mapping	6324
The second F mapping	15840
${L_2}^{-1}$ mapping	1764
Total	24504

F is a map that consists of 2 layers of Oil-Vinegar construction and 24 Oil-Vinegar polynomials. Detailed description of F refers to Section 3.2. The number of multiplications of the components of F is summarized in Table 3.

Table 3. Number of Multiplications in Components of F Mapping

	The first layer	The second layer
V_iO_j	2448	4320
V_iV_j	3672	11160
V_i	204	360
Total	6324	15840

4 Implementations and Experimental Results

4.1 Overview of Our Implementation

Our design is programmed in VHDL by using Quartus II, and implemented on a EP2S130F1020I4 FPGA device, which is a member of ALTERA Stratix family. Table 4 summarizes the performance of our implementation of Rainbow signature measured in clock cycles, which shows that our design takes only 198 clock cycles to generate a Rainbow signature. The experimental results of other key components are described in the following sub-sections.

Table 4. Running Time of Our Implementation

Steps	Components	Clock cycles
1	${L_1}^{-1}$ mapping	5
2	The first round F mapping	45
3	The first round Gaussian elimination	12
4	The second round F mapping	111
5	The second round Gaussian elimination	12
6	${L_2}^{-1}$ mapping	13
	Total	198

4.2 Implementation of Multiplication over Finite Fields

Our multiplier can execute a multiplication over $GF(2^8)$ within 1 clock cycle, similar to the performance of [32]. The running time of a multiplication is 10.768 ns. The compilation report is shown in the second column of Table 5.

Table 5. *FPGA* Implementations of the Multiplier, Inverter and Gaussian Elimination

Components	Multiplier	Inverter	Gaussian elimination
Combinational *ALUTs*	37	227	21718
Dedicated logic registers	0	0	1644
Total pins	24	16	98

4.3 Implementation of Multiplicative Inverse over Finite Fields

Our inverter can execute a multiplicative inverse over $GF(2^8)$ with only 1 clock cycle. The running time of a multiplicative inverse is 24.437 ns. The compilation report is shown in the third column of Table 5.

4.4 Implementation of L_1^{-1}, L_2^{-1} and F Mappings

Since addition operation over $GF(2^8)$ can be executed along with multiplication within one clock cycle, the running of addition can then be ignored. Table 6 shows that the running time of ${L_1}^{-1}$ and ${L_2}^{-1}$ transformations takes 18 clock cycles, and Table 7 illustrates that the running time of F mapping takes 156 clock cycles.

Table 6. Running Time of ${L_1}^{-1}$ and ${L_2}^{-1}$ Mappings

Components	Clock Cycles
L_1 offset	1
${L_1}^{-1}$	4
L_2 offset	1
${L_2}^{-1}$	12
Total	18

Table 7. Running Time of F Mapping

	The first layer	The second layer
V_iO_j	17	30
V_iV_j	26	78
V_i	2	3
Total	45	111

4.5 Implementation of Gaussian Elimination over Finite Fields

The device utilization summary of the Gaussian elimination is shown in Table 5. The number of clock cycles required to solve a 12×12 matrix is 12, which is 0.24 us with a frequency of 50 MHz. The experimental results of multipliers with three elements is shown in Table 8.

Table 8. $FPGA$ Implementations of Designs based on Fig. 3

Components	Original	This Paper
Combinational $ALUTs$	356	354
Actual time (ns)	32.145	26.640

5 Comparison with Related Works

We are going to compare the implementations of Gaussian elimination and Rainbow signature generation with related works.

Since solving the system of linear equations is a core component in the generation of multivariate signature, the improvement of Gaussian elimination will speed up the the signature generation of Rainbow scheme. As explained in Section 3.6, our optimized Gaussian elimination takes only n clock cycles to solve a $n \times n$ system of linear equations. The comparison of Gaussian elimination is given in Table 9 with matrix size 8×8. It is shown that our proposed design has an overwhelming performance in clock cycles.

Table 9. Comparison of Gaussian Elimination with Matrix Size 8×8

Scheme	Clock Cycles
Original Gaussian elimination	128
Linear array based real matrix [29]	82
Systolic Gauss-Jordan algorithm [27]	32
Systolic Gaussian elimination [28]	31
Gaussian elimination over $GF(2^8)$ [26]	16
Implementation in this paper	8

By integrating some optimizations proposed in this paper, we build a new hardware implementation of Rainbow signature. The performance comparison of different signature schemes in clock cycles is given in Table 10. It is shown that our proposed design takes only 198 clock cycles to generate a Rainbow signature, a new record in generating digital signatures and four times faster than the 804-clock-cycle Balasubramanian-Bogdanov-Carter-Ding-Rupp design [14] with similar parameters.

Table 10. Performance Comparison of Signature Schemes

Scheme	Clock cycles
RSA-PSS [6]	348672
EN-TTS [11]	60000
TTS [13]	38000
Rainbow [13]	22000
ECC-163, CMOS [7]	3507
Rainbow [14]	804
Rainbow (42,24), XC3 [7]	623
This paper	198

6 Conclusions

We propose a new optimized hardware implementation of Rainbow signature scheme, which can generate a Rainbow signature with only 198 clock cycles, a new record in generating digital signatures.

Since multiplication, multiplicative inverse and solving linear system of equations over finite fields are playing the key role in generating multivariate signature scheme, their implementations were optimized in this paper. They can also be applied to various applications, such as matrix factorization, matrix inversion, and multivariate PKCs.

Our main contributions include three parts. First, we develop a new parallel hardware design for the Gaussian elimination, which can solve a $n \times n$ linear system of equations with only n clock cycles. Second, we design a novel multiplier to speed up the multiplication of three elements over a finite field. Third, we further optimize the parallelization process of the hardware to generate the Rainbow signature. By integrating these optimizations, we build a new hardware implementation, which takes only 198 clock cycles to generate a Rainbow signature and four times faster than the 804-clock-cycle Balasubramanian-Bogdanov-Carter-Ding-Rupp design [14] with similar parameters.

References

1. F.-X. Standaert, G. Rouvroy, and J.-J. Quisquater. FPGA implementations of the DES and Triple-DES masked against power analysis attacks. In *International Conference on Field Programmable Logic and Applications*, pages 1–4, August 2006.
2. M. McLoone and J.V. McCanny. A high performance FPGA implementation of DES. In *2000 IEEE Workshop on Signal Processing Systems*, pages 374–383, 2000.
3. Chi-Wu Huang, Chi-Jeng Chang, Mao-Yuan Lin, and Hung-Yun Tai. The FPGA implementation of 128-bits AES algorithmbased on four 32-bits parallel operation. In *The First International Symposium on Data, Privacy, and E-Commerce*, pages 462–464, November 2007.

4. Chi-Jeng Chang, Chi-Wu Huang, Hung-Yun Tai, Mao-Yuan Lin, and Teng-Kuei Hu. 8-bit AES FPGA implementation using block RAM. In *33rd Annual Conference of the IEEE Industrial Electronics Society*, pages 2654–2659, November 2007.
5. S.S. Ghoreishi, M.A. Pourmina, H. Bozorgi, and M. Dousti. High speed RSA implementation based on modified Booth's technique and Montgomery's multiplication for FPGA platform. In *Second International Conference on Advances in Circuits, Electronics and Micro-electronics*, pages 86–93, October 2009.
6. Johann. High-speed RSA hardware based on Barrets modular reduction method. In *Cryptographic Hardware and Embedded Systems*, pages 95–136. Springer, 2000.
7. A. Bogdanov, T. Eisenbarth, A. Rupp, and C. Wolf. Time-area optimized public-key engines: $MQ - Cryptosystems$ as replacement for elliptic curves? In *Cryptographic Hardware and Embedded Systems-CHES 2008: 10th International Workshop, Washington, DC, USA, August 10-13, 2008, Proceedings*, page 45. Springer-Verlag New York Inc, 2008.
8. W. Gao and S. Simmons. A study on the VLSI implementation of ECC for embedded DRAM. In *Canadian Conference on Electrical and Computer Engineering*, volume 1, pages 203–206, May 2003.
9. J. Ding and D. Schmidt. Multivariate public key cryptosystems. In *Advances in Information Security*, volume 25. Springer, 2006.
10. B.Y. Yang, C.M. Cheng, B.R. Chen, and J.M. Chen. Implementing minimized multivariate PKC on low-resource embedded systems. *Security in Pervasive Computing*, pages 73–88, 2006.
11. B.Y. Yang, J.M. Chen, and Y.H. Chen. TTS: High-speed signatures on a low-cost smart card. *Cryptographic Hardware and Embedded Systems-CHES 2004*, pages 318–348, 2004.
12. A. Chen, C.H. Chen, M.S. Chen, C.M. Cheng, and B.Y. Yang. Practical-sized instances of Multivariate PKCs: Rainbow, TTS, and ℓic-derivatives. *Post-Quantum Cryptography*, pages 95–108, 2008.
13. A. Chen, M.S. Chen, T.R. Chen, C.M. Cheng, J. Ding, E. Kuo, F. Lee, and B.Y. Yang. SSE implementation of multivariate PKCs on modern x86 CPUs. *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 33–48, 2009.
14. S. Balasubramanian, A. Bogdanov, A. Rupp, J. Ding, and H.W. Carter. Fast multivariate signature generation in hardware: The case of Rainbow. In *16th International Symposium on Field-Programmable Custom Computing Machines*, pages 281–282, April 2008.
15. S. Talapatra and H. Rahaman. Low complexity montgomery multiplication architecture for elliptic curve cryptography over $GF(p^m)$. In *VLSI System on Chip Conference (VLSI-SoC), 2010 18th IEEE/IFIP*, pages 219–224, September 2010.
16. G. Perin, D.G. Mesquita, F.L. Herrmann, and J.B. Martins. Montgomery modular multiplication on reconfigurable hardware: Fully systolic array vs parallel implementation. In *2010 VI Southern Programmable Logic Conference*, pages 61–66, March 2010.
17. S. Talapatra, H. Rahaman, and S.K. Saha. Unified digit serial systolic montgomery multiplication architecture for special classes of polynomials over $GF(2^m)$. In *13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, pages 427–432, September 2010.
18. A.H. Namin, K. Leboeuf, R. Muscedere, H. Wu, and M. Ahmadi. High-speed hardware implementation of a serial-in parallel-out finite field multiplier using reordered normal basis. *Circuits, Devices Systems, IET*, 4(2):168–179, March 2010.

19. A.H. Namin, K. Leboeuf, R. Muscedere, Huapeng Wu, and M. Ahmadi. High speed VLSI implementation of a finite field multiplier using redundant representation. In *European Conference on Circuit Theory and Design*, pages 161–164, August 2009.
20. A. Mahboob and N. Ikram. Faster polynomial basis finite field squaring and inversion for $GF(2^m)$ with cryptographic software application. In *International Symposium on Biometrics and Security Technologies*, pages 1–6, April 2008.
21. N. Jachimiec, N. Iliev, and J. Stine. Strategies for VLSI implementations of finite field inversion algorithms. In *48th Midwest Symposium on Circuits and Systems*, volume 2, pages 1589–1592, August 2005.
22. Yang Xiao-hui, Qin Fan, Dai Zi-bin, and Zhang yong fu. Scalable and unified hardware architecture for montgomery inversion computation in GF(p) and $GF(2^n)$. In *IEEE 8th International Conference on ASIC*, pages 843–846, October 2009.
23. D.W. Matula, A. Fit-Florea, and M.A. Thornton. Table lookup structures for multiplicative inverses modulo 2^k. In *17th IEEE Symposium on Computer Arithmetic*, pages 156–163, June 2005.
24. J.H. Jeng. Normal basis inversion in some finite fields. In *Proceedings of the Fifth International Symposium on Signal Processing and Its Applications*, volume 2, pages 701–703, 1999.
25. A. Bogdanov, MC Mertens, C. Paar, J. Pelzl, and A. Rupp. A parallel hardware architecture for fast Gaussian elimination over GF(2). In *14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 237–248. IEEE, 2006.
26. S.R. Balasubramanian. A parallel hardware architecture for fast signature generation of Rainbow. In *the Department of Electrical and Computer Engineering of the College of Engineering, University of Cincinnati, Master thesis*, 2007.
27. Chin-Liang Wang and Jung-Lung Lin. A systolic architecture for computing inverses and divisions in finite fields $GF(2^m)$. *IEEE Transactions on Computers*, 42(9):1141–1146, September 1993.
28. B. Hochet, P. Quinton, and Y. Robert. Systolic Gaussian elimination over GF(p) with partial pivoting. *IEEE Transactions on Computers*, 38(9):1321–1324, September 1989.
29. G. Govindu, S. Choi, V. Prasanna, V. Daga, S. Gangadharpalli, and V. Sridhar. A high-performance and energy-efficient architecture for floating-point based LU decomposition on FPGAs. In *18th International Parallel and Distributed Processing Symposium*, page 149. IEEE, 2004.
30. J. Ding, B.Y. Yang, C.H.O. Chen, M.S. Chen, and C.M. Cheng. New differential-algebraic attacks and reparametrization of Rainbow. In *Proceedings of the 6th international conference on Applied cryptography and network security*, pages 242–257. Springer-Verlag, 2008.
31. M.R. Schroeder and MR Schroeder. *Number theory in science and communication*. Springer, 1986.
32. C.C. Wang, TK Troung, H.M. Shao, L.J. Deutsch, J.K. Omura, and I.S. Reed. VLSI architectures for computing multiplications and inverses in $GF(2^m)$. *IEEE Transactions on Computers*, pages 709–717, 1985.