High-speed Hardware Implementation of Rainbow Signature on FPGAs

Shaohua Tang, Haibo Yi, Jintai Ding,

Huan Chen, and Guomin Chen

South China Univ of Tech

csshtang@scut.edu.cn

Outline

- Introduction
- Background
- Proposed Hardware Design for Rainbow Signature
- Implementations and Experimental Results
- Comparison with Related Work
- Conclusions

Introduction

- The Oil-Vinegar family of Multivariate Public Key Cryptosystems consists of three families:
  - balanced Oil-Vinegar
  - unbalanced Oil-Vinegar
  - Rainbow
    - a multilayer construction using unbalanced Oil-Vinegar at each layer
- There have been some previous works to efficiently implement multivariate signature schemes, e.g.,
  - TTS on a low-cost smart card
  - minimized multivariate PKC on low-resource embedded systems
  - some instances of MPKCs
  - SSE implementation of multivariate PKCs on modern x86 CPUs

Introduction

- Currently the best hardware implementations of Rainbow signature are:
  - A parallel hardware implementation of Rainbow signature [8]
    - the fastest work (not best in area utilization),
    - which takes 804 clock cycles to generate a Rainbow signature;
  - A hardware implementation of multivariate signatures using systolic arrays [9],
    - which optimizes in terms of certain trade-off between speed and area.

[8] S. Balasubramanian, et al. *Fast multivariate signature generation in hardware: The case of Rainbow*. FPCC 2008.

[9] A. Bogdanov, et al. *Time-area optimized public key engines: MQ Cryptosystems as replacement for elliptic curves?* CHES 2008.

Introduction

- The major computation components in generation of Rainbow signature include:
  - Multiplication of elements in finite field;
  - Multiplicative inversion of elements in finite fields;
  - Solving system of linear equations over finite fields.

- Therefore, we focus on further improvement in these three directions.

Our Focus and Contributions

- The **focus** of our work
  - to further speed up hardware implementation of Rainbow signature generation
  - without consideration of the area cost
- Our **contributions**:
  - the improvement of the multiplication over finite fields;
  - the development of a new parallel hardware design for the Gauss-Jordan elimination to solve a $n \times n$ system of linear equations with only n clock cycles;
  - the design of a new partial multiplicative inverter;
  - other minor optimizations of the parallelization process.

Overview of Rainbow Signature Scheme

- 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

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

Background

Overview of Rainbow Signature Scheme (continued)

- Private key
  - Two randomly chosen invertible affine linear transformations L_1 and L_2

$$L_1 : k^{n-v_1} \rightarrow k^{n-v_1}$$

$$L_2 : k^n \rightarrow k^n$$

  - The central mapping F
    - F has u-1 layers of Oil-Vinegar construction
    - The l-th layer: o_l polynomials
      - Oil variables: $\{x_i \mid i \in O_l\}$
      - Vinegar variables: $\{x_j \mid j \in S_l\}$

Overview of Rainbow Signature Scheme (continued)

- Public key
  - The finite field *k*
  - The n-v_1 polynomial components of

$$\bar{F} = L_1 \circ F \circ L_2$$

- Signature generation
  - The message: $Y = (y_1, \ldots, y_{n-v_1}) \in k^{n-v_1}$
  - The signature is derived by computing

$$\bar{F}^{-1} = L_2^{-1} \circ F^{-1} \circ L_1^{-1}(Y)$$

Overview of Rainbow Signature Scheme (continued)

- *Signature generation*
  1. Compute

$$\bar{Y}' = L_1^{-1}(Y)$$

  2. To solve the equation

$$F(X) = \bar{Y}'$$

and obtain a solution

$$\bar{X} = (\overline{x_1}, \ldots, \overline{x_n})$$

satisfying

$$F(\overline{X}) = \bar{Y}'$$

Overview of Rainbow Signature Scheme (continued)

- *Signature generation*

  3. Compute

$$X' = L_2^{-1}(\bar{X}) = (x_1', \ldots, x_n')$$

  – Then X' is the signature for message Y.

- Signature verification
  – Suppose the signature X'
  – Compute $\bar{F}(X') = Y'$
  – If $Y' = Y$ holds, the signature is accepted; otherwise, rejected.

Parameters of Rainbow Adopted in Our Work

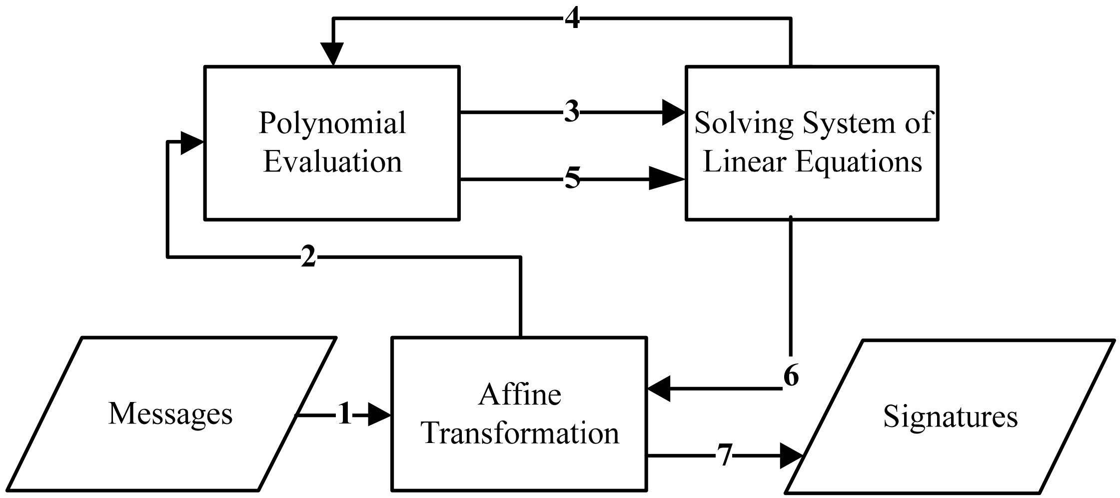
– Suggested in [14], security level above 2^{80}.

Parameter	Rainbow
Ground field	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)

[14] J. Ding, B.Y. Yang, C.H.O. Chen, M.S. Chen, and C.M. Cheng. *New differential-algebraic attacks and reparametrization of Rainbow*. ACNS 2008, pp. 242-257

Proposed Hardware Design for Rainbow Signature

- **Overview of our Hardware Design**
  - Flowchart to generate Rainbow signature:



  - Computing affine transformations, L_1^{-1} and L_2^{-1}.
  - Evaluating multivariate polynomials in F maps.
  - Solving system of linear equations.

Choice of Irreducible Polynomials

- The choice of the irreducible polynomials for the finite field is a critical part of our hardware design, since
  - it determines the structure of the finite field,
  - and affects the efficiency of the operationsover the finite field.
- The irreducible polynomials for GF(2^8) can be expressed as 9-bit binary digits with the form $x^8 + x^k + ... + 1$, where 0 < k < 8.
  - There are totally 16 candidates.
- We evaluate the performance of the multiplications based on these irreducible polynomials respectively.
  - By comparing the efficiency of signature generations basing on different irreducible polynomials,

$$x^8 + x^6 + x^3 + x^2 + 1$$

is finally chosen.

Efficient Design of Multiplication of Three Elements

- In Rainbow signature generation, we notice that
  - there exist not only multiplication of two elements
  - but also multiplication of three elements
  - for example:
    - the evaluation of Oil-Vinegar polynomials

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

- Let **ThreeMult(v1,v2,v3)** stand for multiplication of three elements, where v1, v2, v3 are operands.

Efficient Design of Multiplication of Three Elements

- The new design is based on a new observation that,
  - in multiplication of elements over GF(2^8), it is

    much **faster** to multiply everything first then perform modular operation

    than the other way around.

$$d(x) = a(x) \times b(x) \times c(x)(\mathrm{mod}(f(x))) = \sum_{i=0}^{7} d_i x^i$$

  - This is quite anti-intuitive, and it works only over small fields.
  - This idea, in general, is not applicable for large fields.
- Therefore, we design new implmentation to speedup multiplication of three elements.

Multiplicative Inversion and Partial Multiplicative Inversion

- The multiplicative inverse over the finite field is a crucial but time-consuming operation in multivariate signature.
- An optimized design of the inverter can really help to imporve the overall performance.
- Suppose f(x) is the irreducible polynomial and β is an element over GF(2^8), according to the Fermat's theorem, we have $\beta^{2^8} = \beta,\ \text{and}\ \beta^{-1} = \beta^{2^8-2} = \beta^{254}.$
- Since $2^8 - 2 = 2 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7,$

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

Multiplicative Inversion and Partial Multiplicative Inversion

- We adopt the three-input multiplier to design the partial inverter.
- Note that $\beta^{-1} = \beta^2 \beta^4 \beta^8 \beta^{16} \beta^{32} \beta^{64} \beta^{128}$,

  and $\beta^{-1} = ThreeMult(S_1, S_2, \beta^{128})$,

  - where **ThreeMult(v1,v2,v3)** stands for multiplication of three elements, where v1, v2, v3 are operands.
  - Let $S_1 = ThreeMult(\beta^2, \beta^4, \beta^8)$,

    $S_2 = ThreeMult(\beta^{16}, \beta^{32}, \beta^{64})$
  - We call the triple

$$(S_1, S_2, \beta^{128})$$

the partial multiplicative inversion of β.

Solving System of Linear Equations

Algorithm 1 Solving a system of linear equations

Ax = b with 12 iterations, where A is a 12 × 12 matrix

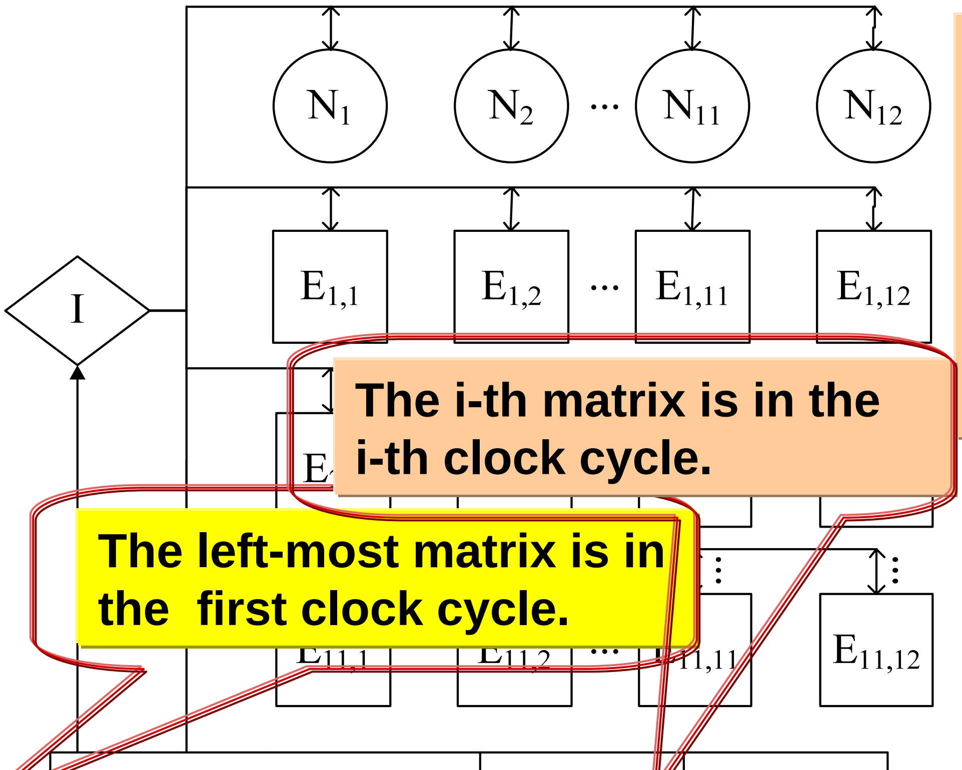
```
var
    i: Integer;
begin
    i := 0;
    Pivoting(i = 0);
    repeat
        Partial_inversion(i), Normalization(i), Elimination(i);
        Pivoting(i+1);
         i:= i+1;
    until i = 12
end.
```

the optimized Gauss-Jordan elimination with 12 iterations, which consists of pivoting, partial multiplicative inversion, normalization and elimination in each iteration.

They are designed to perform simultaneously.

it takes only one clock cycle to perform one iteration.

- The architecture for solving system of linear equations.



There exist three kinds of cells in the architecture:
I: partial multiplicative inversion;
Ni: normalization;
Eij: elimination;

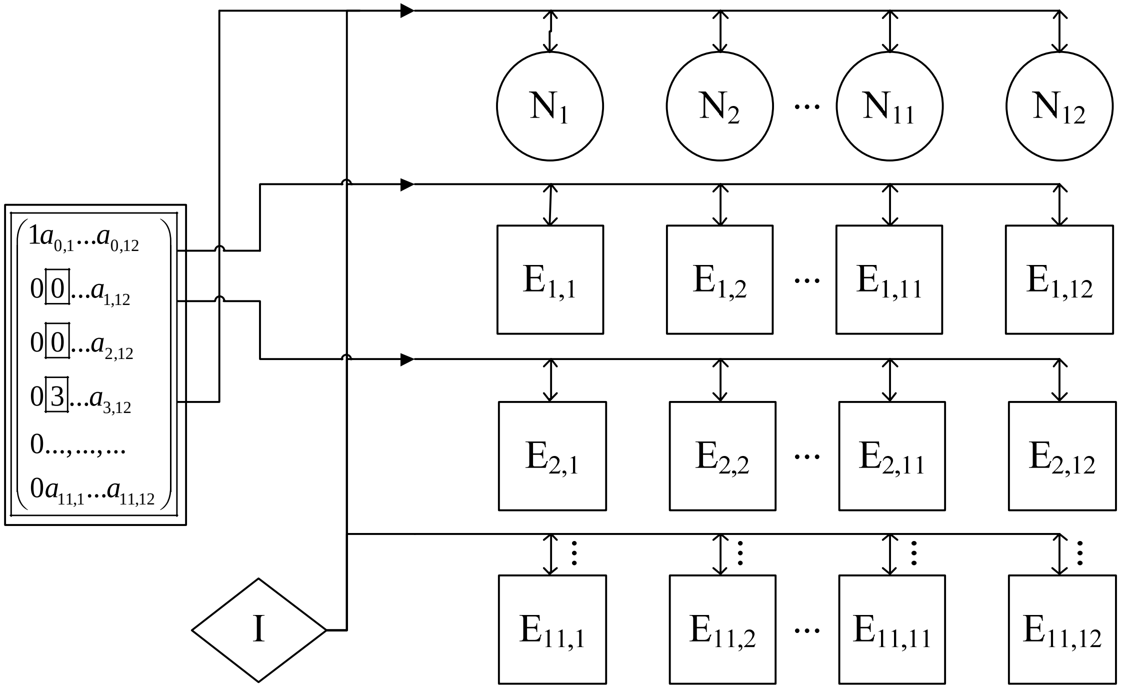
Totally, 1 I, 12 Ni, 132 Eij cells.

The matrixes below are used to illustrate how the matrix changes in each clock cycle.

$$\begin{pmatrix} a_{0,0}a_{0,1}\ldots a_{0,11}a_{0,12} \\ a_{1,0}a_{1,1}\ldots a_{1,11}a_{1,12} \\ \ldots,\ldots,\ldots,\ldots,\ldots,\ldots \\ a_{11,0}a_{11,1}\ldots a_{11,11}a_{11,12} \end{pmatrix} \begin{pmatrix} 1a_{0,1}'\ldots 0a_{0,11}'a_{0,12}' \\ 0a_{1,1}'\ldots 0a_{1,11}'a_{1,12}' \\ 0\ldots,\ldots,\ldots,\ldots,\ldots,\ldots \\ 0a_{11,1}'\ldots 0a_{11,11}'a_{11,12}' \end{pmatrix} \cdots \cdots \begin{pmatrix} 10\ldots 0a_{0,11}''a_{0,12}'' \\ 01\ldots 0a_{1,11}''a_{1,12}'' \\ 00\ldots,\ldots,\ldots,\ldots \\ 00\ldots 0a_{11,11}''a_{11,12}'' \end{pmatrix} \begin{pmatrix} 10\ldots 00a_{0,12}''' \\ 01\ldots 00a_{1,12}''' \\ 0\ldots,\ldots,\ldots,\ldots \\ 00\ldots 01a_{11,12}''' \end{pmatrix}$$

Solving System of Linear Equations

❖ Pivoting operation



In each clock cycle, the pivot element is sent to I cell for partial multiplicative inversion.

The pivot row is sent to Ni for normalization.

The other rows except the pivot row are sent to Eij for elimination.

Then, I, Ni, and Eij cells can execute in parallel.

Example: before the second iteration,

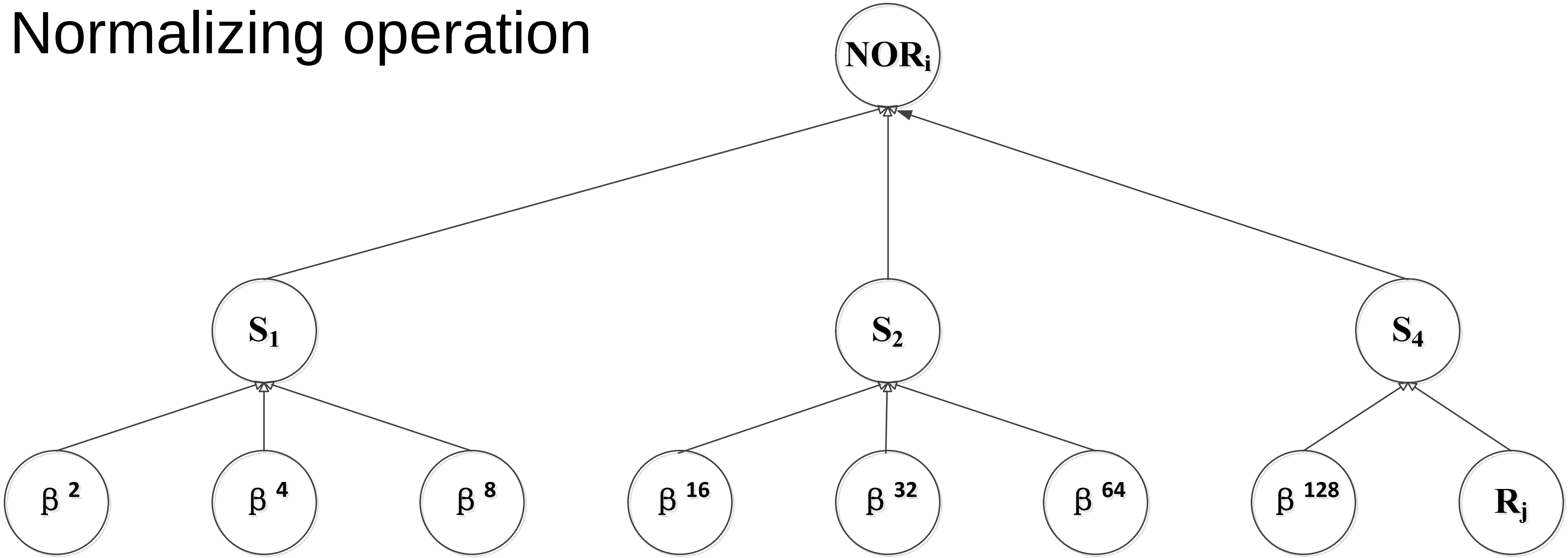
The second row is the pivot row, but the pivot element is zero.

The fourth row can be chosen as the new pivot row since a_{31} is nonzero.

Then a_{31} is sent to I cell, the fourth row is sent to Ni, the other rows are sent to Eij.

The computation of one iteration can be performed with one clock cycle.

- Normalizing operation



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

$$S_1 = ThreeMult(\beta^2, \beta^4, \beta^8),$$

$$S_2 = ThreeMult(\beta^{16}, \beta^{32}, \beta^{64})$$

$$S_4 = TwoMult(\beta^{128}, R_i)$$

$$NOR_i = ThreeMult(S_1, S_2, S_4)$$

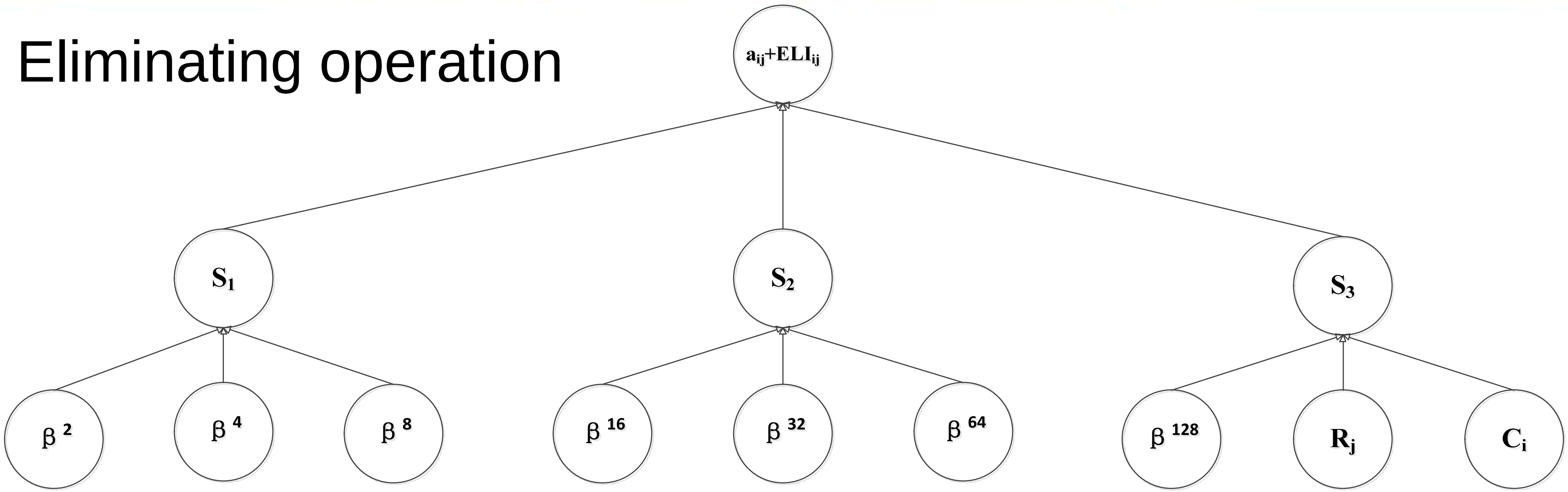
S1 and S2 are executed in I cell.

S4 and NORi are executed in Ni cell.

S1, S2 and S4 can be implemented in parallel in each iteration.

(R_i: the i-th element in the pivot row;)

- Eliminating operation



S1 and S2 are executed in I cell.

S3 and ELIij are executed in Eij cell.

S1, S2 and S3 can be implemented in parallel in each iteration.

$$S_1 = ThreeMult(\beta^2, \beta^4, \beta^8),$$

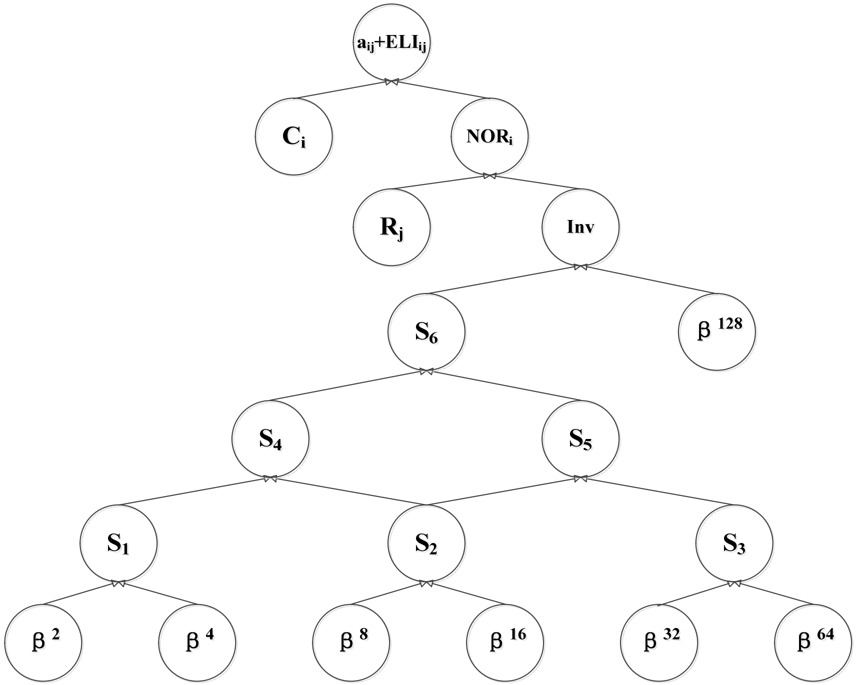
$$S_2 = ThreeMult(\beta^{16}, \beta^{32}, \beta^{64})$$

$$S_3 = ThreeMult(\beta^{128}, R_j, C_i)$$

$$ELI_{ij} = a_{ij} + ThreeMult(S_1, S_2, S_3)$$

(R_j: the j-th element in the pivot row;
Ci: the i-th element in the pivot column;)

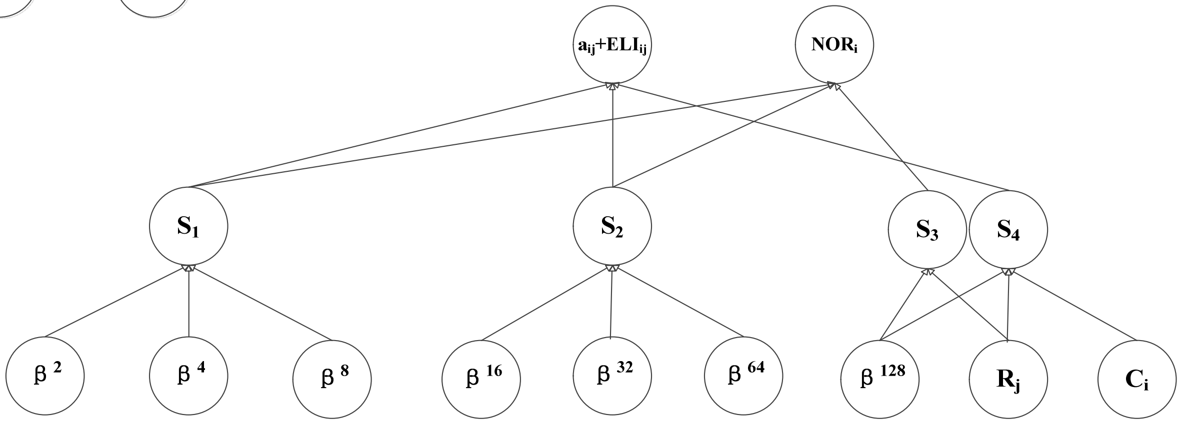
- Original design VS Optimized design



The critical path of the original Gauss-Jordan elimination is five.

The critical path of our design is two.

Therefore, our optimization reduce the critical path from five to two.



Affine Transformations and Polynomial Evaluations

- Two affine Transformations

$$L_1^{-1} : k^{24} \to k^{24}, L_2^{-1} : k^{42} \to k^{42}$$

are computed by invoking vector addition and matrx-vector multiplication.

- Two-layer Oil-Vinegar constructions include 24 Oil-Vinegar polynomials that are evaluated by invoking multiplication and addition.

The Oil-Vinegar polynomial:

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

Table 2 Number of multiplications in affine transformations and polynomial evaluations

Components	Number of multiplications
L_1^{-1} transformation	576
The first 12 polynomial evaluations	6324
The second 12 polynomial evaluations	15840
L_2^{-1} transformation	1764
Total	24504

Table 3 Number of Multiplications in Components of Polynomial Evaluations

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

Implementations and Experimental Results

- Our design is programmed in **VHDL**
  - and implemented on a EP2S130F1020I4 **FPGA device**,
  - which is a member of ALTERA Stratix II family.
- All the experimental results mentioned in this section are extracted after place and route.
- 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.
  - In other words, our implementation takes 3960 ns to generate a Rainbow signature with the frequency of 50 MHz.

Table 4 Running time of our implementation in clock cycles

Step No.	Components	Clock cycles
1	L_1^{-1} transformation	5
2	The first 12 polynomial evaluations	45
3	The first round of solving system of linear equations	12
4	The second 12 polynomial evaluations	111
5	The second round of solving system of linear equations	12
6	L_2^{-1} transformation	13
	Total	198

Table 5 FPGA implementations of the multiplier, partial inverter, Gauss-Jordan elimiation

Components	Combinational ALUTs	Dedicated logic resisters	Clock cycles	Running time (ns)
Multiplier	37	0	1	10.768
Partial inverter	22	0	1	9.701
Gauss-Jordan elimination	21718	1644	12	240

(with a frequency of 50 MHz)

Table 6 The resource consumptions for each cell in the architecture for solving system of linear equations

Cell	Used for	Two-input multiplier	Three-input multiplier	Adder
I cell	Partial inversion	0	2	0
N cell	Normalization	1	1	0
E cell	Elimination	0	2	1

Table 7 Clock cycles and running time of two affine transformations

Components	Clock cycles	Running time (ns)
L_1 offset	1	20
L_1^{-1}	4	80
L_2 offset	1	20
L_2^{-1}	12	240
Total	18	360

(with a frequency of 50 MHz)

Table 8 Clock cycles and running time of polynomial evaluations

Components	V_iO_j	V_iV_j	V_i	Total cycles	Total time (ns)
The first layer	17	26	2	45	900
The second layer	30	78	3	111	2220

(with a frequency of 50 MHz)

Table 9 Comparison of solving system of linear equations with matrix size 12 × 12

Scheme	Clock cycles
Original Gauss-Jordan elimination	1116
Original Gaussian elimination	830
Wang-Lin's Gauss-Jordan elimination [12]	48
B. Hochet's Gaussian elimination [13]	47
A Bogdanov's Gaussian elimination [11]	24
Implementaion in this paper	12

Table 10 Performance comparison of signature schemes

Scheme	Clock cycles
en-TTS [5]	16000
Rainbow (42, 24) [9]	3150
Long-message UOV [9]	2260
Rainbow [8]	804
Short-message UOV [9]	630
This paper	198

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,
  - four times faster than the 804-clock-cycle implementation in [8],
- Our main contributions include three parts
  - First, we develop a new parallel hardware design for the Gauss-Jordan elimination, and solve a 12×12 system of linear equations with only 12 clock cycles.
  - Second, a novel multiplier is designed to speed up multiplication of three elements over finite fields.
  - Third, we design a novel partial multiplicative inverter to speed up the multiplicative inversion of finite field elements.

Conclusions

- Note that our implementation focuses solely on speeding up the signing process,
  - in terms of area, we compute the size in gate equivalents (GEs), about 150,000 GEs,
  - which is 2-3 times the area of [8].

[8] S. Balasubramanian, et al. *Fast multivariate signature generation in hardware: The case of Rainbow*. FPCC 2008.

Thank you

- ✓ Contact us via email:

  shtang@ieee.org
  csshtang@scut.edu.cn