PYRIT: Polynomial Ring Transforms for Fast Erasure Coding

Jonathan Detchart, Jérôme Lacan
ISAE-Supaéro, Université de Toulouse, France

In today's storage systems, erasure codes are widely used and provide excellent reliability to failures. But in return, this kind of technique is limited by the complexity of the arithmetic used. Most of the complexity of erasure codes consist in making linear combinations over a finite field.

Multiplication of field elements can be done using the xor-based representation or a lookup table (LUT). Thanks to special SIMD instructions, LUT is, in most cases the fastest implementation.

We propose the first erasure code based on ring structures: PYRIT (PolYnomial RIng Transform). Our solution replaces the multiplication in a finite field by the multiplication in a bigger ring, and uses special transforms between fields and rings. Making multiplications into a ring allows to reduce the complexity of the coding and the decoding processes. This also allows some optimizations which are not possible when using a classic `xor` based implementation (with multiplication in a field).

Rather than multiplying elements of a finite field, we use isomorphic functions between a field and a bigger ring to transform each field element into a ring element using the following properties:

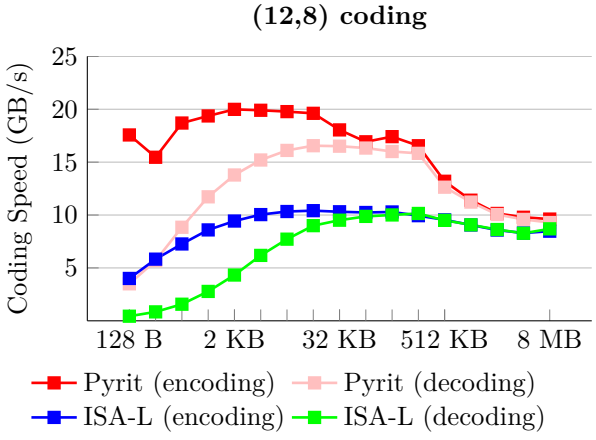
The finite field with 2^m elements has the form $\mathbb{F}_{2^m} = \mathbb{F}_2[x]/(p(x))$ where $p(x)$ is an irreducible binary polynomial of degree m.

The polynomial $p(x)$ is necessarily a divisor of a polynomial x^n+1. Let $R_{2,n}$ denote the ring $R_{2,n} = \mathbb{F}_2[x]/(x^n+1)$. It can be shown that the ideal $A = ((x^n+1)/p(x))$ of $R_{2,n}$ is isomorphic to $\mathbb{F}_{2^m}$. This means that working in a field is (mathematically) equivalent to working in a (ideal of the) ring.

For polynomials $p(x)$ which have an All-One structure, like *e.g.* $1+x+x^2+x^3+x^4$, or an Equally-Space structure like *e.g.* $1+x^3+x^6$, we propose three methods to make the correspondence between a field and a ring: (1) Embedding, where we just consider a field element as a ring element, (2) Parity, which consists in adding a parity bit, and (3) Sparsest representation, which chooses the sparsest ring element corresponding to the field element. By extending [2], we apply these functions to the different types of data of the coding process: source data, encoding/decoding matrices and repair data.

Thanks to the ring structure, the `xor`-based representation of elements is only composed of cyclic diagonals. Moreover, sparse coding matrices can be obtained by choosing the adequate field-to-ring transform. This significantly decreases both encoding and decoding complexities.

By fully exploiting these two properties, we propose a simple but fast `xor`-based implementation based on ring operations. Indeed, by unrolling the `xor` operations, the encoding and decoding speeds can be improved by up to 200% compared to the best known implementations [1]. The figure below shows the performance of a code with parameters $(n,k) = (12,8)$ (*i.e.* 12 encoding blocks from 8 source blocks) on an Intel Skylake i6500 3.2 GHz with 16 GB RAM).



References

[1] ISA-L: Intel storage acceleration library. In *https://01.org/intel-storage-acceleration-library-open-source-version*.

[2] Itoh, T., and Tsujii, S. Structure of parallel multipliers for a class of fields GF(2^m). *Information and Computation 83*, 1 (1989), 21 – 40.

PYRIT: Polynomial Ring Transforms for Fast Erasure Coding

Jonathan DETCHART, Jérôme LACAN
ISAE-Supaéro, Université de Toulouse, France

Context

Erasure codes are used in storage systems to cope with failures. They are based on linear combinations carried out in finite fields.
Problem: multiplications in a finite field are complex.

Objective

Make coding operations as fast as possible by making multiplications in a ring.

⇓

Proposal

Pyrit (PolYnomial RIng Transform): the first erasure code based on ring structures.

1. Moving from a field to a bigger ring.
2. Making multiplications in this ring.
3. Going back from a ring to a field.

Correspondence between a field and a ring

For n odd, if $x^n-1 = \Pi_{i=1}^{l} p_i(x)$, where p_i are irreducible polynomials, the ring $R_{2,n} = \mathbb{F}_2[x]/(x^n+1)$ is equal to the direct sum of the minimal ideals $A_i = ((x^n+1)/p_i(x))$ for $i = 1, \ldots, l$.

The finite field $\mathbb{F}_{2^m} = \mathbb{F}_2[x]/(p_i(x))$, where m is the degree of $p_i(x)$, is isomorphic to the ideal A_i.

⇒ working in a field is (mathematically) equivalent to working in a (ideal of the) ring.

For polynomials $p_i(x)$ which have the property:

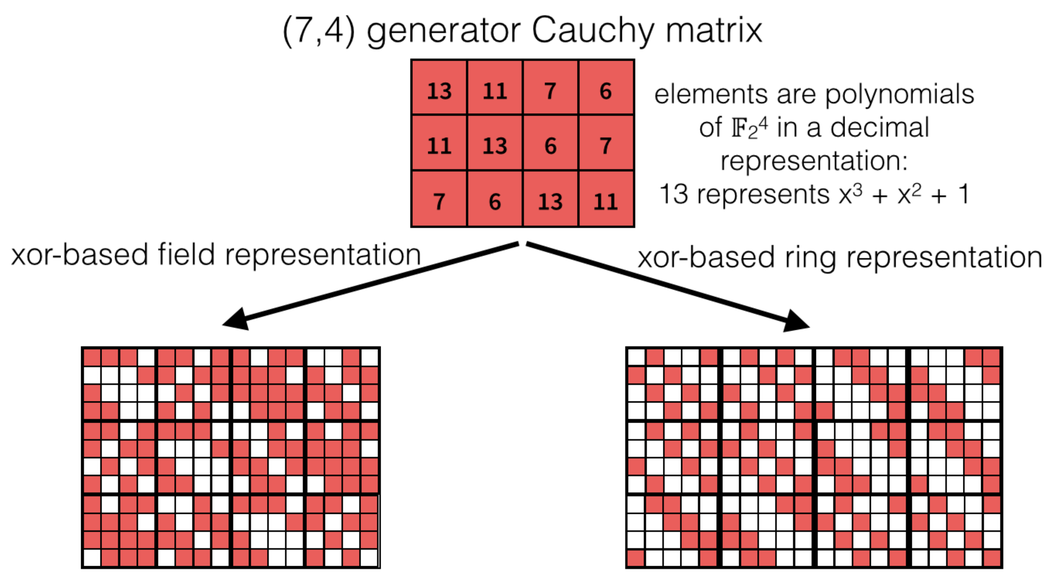
- AOP (All-one polynomials) e.g. $1 + x + x^2 + x^3 + x^4$
- ESP (Equally Spaced polynomials), e.g. $1 + x^3 + x^6$

We propose three methods to make the correspondence between a field and a ring:

- Embedding: just consider a field element as a ring element
- Parity: add a parity bit
- Sparsest representation: choose the sparsest ring element corresponding to the field element

We apply these functions for the different types of data of the coding process: source data, encoding/decoding matrices and repair data. This can be seen as an extension of [2].

Field vs ring xor-based representations



Improving processing speed

ring operations are faster than field ones:

- Number of xor operations: the sparsest method builds low-density matrices (gain of 18% above)
- Data organization: the modulo is just a cyclic shift: the elements are only composed by cyclic diagonals:
  - smaller representation in memory
  - less branches in the code
  - more unrolling
- Easy scheduling: thanks to the cyclic representation of the matrix elements

These properties allowed us to build a simple and fast C implementation.

Coding and decoding speeds are compared to [1] (the fastest implementation we know for erasure codes) for parameters $(n, k) = (12, 8)$ (*i.e.* 12 encoding blocks from 8 source blocks) and $(60, 40)$ in figure below.
Hardware: Intel CPU Skylake i6500 3.2 GHz with 16 GB DRAM.
Our implementation can increase **speed** by **more than 200 %**.

References

[1] ISA-L: Intel storage acceleration library.
In *https://01.org/intel-storage-acceleration-library-open-source-version*.

[2] Toshiya Itoh and Shigeo Tsujii.
Structure of parallel multipliers for a class of fields GF(2^m).
Information and Computation, 83(1):21 – 40, 1989.