Towards optimal Toom-Cook-3 multiplication

for univariate binary polynomials

Marco Bodrato

Centro "Vito Volterra" – Università di Roma "Tor Vergata" – Italy

WAIFI 2007 - June 21^{th}

see appendices

Polynomial multiplication in $\mathrm{GF}(2)[x]$

The problem

We start from two dense binary polynomials

$$u, v \in \mathrm{GF}(2)[x]$$

and we need the product

$$w = u \cdot v \in \mathrm{GF}(2)[x]$$

Assume monomial base.

$$\begin{aligned} u =& \quad x^{d_u} \ldots 0 \cdot x^6 + 1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x + 1 \\ v =& \quad x^{d_v} \ldots 1 \cdot x^6 + 0 \cdot x^5 + 0 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x + 0 \\ \rightsquigarrow w =& \; x^{d_u + d_v} \ldots 1 \cdot x^6 + 1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x + 0 \end{aligned}$$

Polynomial multiplication in $\mathrm{GF}(2)[x]$

The problem

We start from two dense binary polynomials

$$u, v \in \mathrm{GF}(2)[x]$$

and we need the product

$$w = u \cdot v \in \mathrm{GF}(2)[x]$$

Compact dense representation, each bit store a coefficient.

$$\begin{aligned} u &= \qquad\quad [1 \ldots 0110111] \\ v &= \qquad\quad [1 \ldots 1000110] \\ \rightsquigarrow w &= [1 \ldots\ldots\ldots\ldots 1110010] \end{aligned}$$

Polynomial multiplication algorithms

Many algorithms are known for polynomial multiplication.

- Naïve $\mathrm{O}(d^2)$

Each one has a different complexity, and a different range where it is the fastest.

see thresholds

Polynomial multiplication algorithms

Many algorithms are known for polynomial multiplication.

- Naïve $\mathrm{O}(d^2)$
- Karatsuba (Toom-2) (1962) $\mathrm{O}(d^{\log_2 3})$

Each one has a different complexity, and a different range where it is the fastest.

see thresholds

Polynomial multiplication algorithms

Many algorithms are known for polynomial multiplication.

- Naïve — $O(d^2)$
- Karatsuba (Toom-2) (1962) — $O(d^{\log_2 3})$
- Schönhage-FFT (1977) — $O(d \log d \log \log d)$

Each one has a different complexity, and a different range where it is the fastest.

see thresholds

Polynomial multiplication algorithms

Many algorithms are known for polynomial multiplication.

- Naïve — $\mathrm{O}(d^2)$
- Karatsuba (Toom-2) (1962) — $\mathrm{O}(d^{\log_2 3})$
- Toom-Cook-k (1963) — $\mathrm{O}(d^{\log_k 2k-1})$
- Schönhage-FFT (1977) — $\mathrm{O}(d \log d \log \log d)$

Each one has a different complexity, and a different range where it is the fastest.

see thresholds

Some authors say: "Toom's strategy is impossible for $\mathrm{GF}(2)[x]$".
I say: "It is possible and practical"

Recall on Toom-k algorithm

5 phases

1. Splitting

Phase 1, choose a base, homogenise see unbalanced

Choose a base $Y = x^b$ suitable to represent operands with k parts.

$$\begin{array}{lllllll} \mathrm{GF}(2)[x] & & \rightsquigarrow & & & & \mathrm{GF}(2)[x][y,z] \\ u = & [\ldots\ldots\ldots] & \rightsquigarrow & [\ldots]\cdot y^2 + & [\ldots]\cdot yz & +[\ldots]\cdot z^2 & = \mathfrak{u} \\ v = & [\ldots\ldots\ldots] & \rightsquigarrow & [\ldots]\cdot y^2 + & [\ldots]\cdot yz & +[\ldots]\cdot z^2 & = \mathfrak{v} \end{array}$$

Recall on Toom-k algorithm

5 phases

1. Splitting: choose a base, homogenise
2. Evaluation

Phase 2, some linear algebra

Evaluate polynomials $\mathfrak{u}, \mathfrak{v}$ in $2k-1$ different points $(\alpha_i, \beta_i) \in \mathrm{GF}(2)[x]^2$, not just in $\mathrm{GF}(2)$!
Obtain this multiplying a (non square) Vandermonde matrix by the vector of coefficients.

Recall on Toom-k algorithm

5 phases

1. Splitting: choose a base, homogenise
2. Evaluation: $2\times$ matrix-vector multiplication
3. Multiplication

Phase 3, recursive application

▸ see unbalanced

Compute evaluation of the product by multiplying evaluations.
$\mathfrak{w}(\alpha_i, \beta_i) = \mathfrak{u}(\alpha_i, \beta_i) \cdot \mathfrak{v}(\alpha_i, \beta_i)$
Degree $k-1 \times$ degree $k-1 \rightsquigarrow$ degree $2k-2$.
k parts $\times$ k parts $\rightsquigarrow 2k-1$ parts. $\Rightarrow 2k-1$ multiplications.

Recall on Toom-k algorithm

5 phases

1. Splitting: choose a base, homogenise
2. Evaluation: $2\times$ matrix-vector multiplication
3. Multiplication: $(2k-1)\times$ recursive application
4. Interpolation

Phase 4, some more linear algebra

Interpolate to obtain coefficient of the product polynomial.

Obtain this multiplying the inverse of a (square) Vandermonde matrix by the vector of evaluations.

Recall on Toom-k algorithm

5 phases

1. Splitting: choose a base, homogenise
2. Evaluation: $2\times$ matrix-vector multiplication
3. Multiplication: $(2k-1)\times$ recursive application
4. Interpolation: *inverse matrix*-vector multiplication
5. Recomposition

Phase 5, last details

We computed the product in $\mathrm{GF}(2)[x][y,z]$.
Go back to $\mathrm{GF}(2)[x]$ with an evaluation:
$u \cdot v = \mathfrak{u}(Y,1)\mathfrak{v}(Y,1) = \mathfrak{w}(Y,1) = w \in \mathrm{GF}(2)[x]$
where Y, is the “base” chosen during phase 1.

Recall on Toom-k algorithm

5 phases

1. Splitting: choose a base, homogenise
2. Evaluation: $2\times$ matrix-vector multiplication
3. Multiplication: $(2k-1)\times$ recursive application
4. Interpolation: *inverse matrix*-vector multiplication
5. Recomposition: shift and add.

Phase 2 and 4, are critical

Splitting order k gives number $(2k-1)$ of multiplication in phase 3, and asymptotic behaviour $\mathrm{O}(d^{\log_k 2k-1})$. Rigidly.
The choice of evaluation/interpolation points and operation sequences for phases 2 and 4 gives the hidden constant.

Operations we count on for linear algebra

Basic on long operands	(cost)
• Addition(Subtraction)	(`add`) linear
• Mul/div by x^n (optimised with shift)	(`shift`) linear
• Multiplication by a "small" operand	(`Smul`) linear
• Exact division by a "small" operand	(`Sdiv`) linear

"small" actually means fixed: asymptotically small. Typically fits in 1 BYTE.

Composite

- linear combination $l_i \leftarrow (c_j \cdot l_j + c_k \cdot l_k)/d_i$, may be $i = j$
  c_j, c_k, d_i are "small" constants.

Evaluation is Matrix-vector multiplication

After splitting, operands are quadratic polynomials

$$u(y,z) = U_2 y^2 + U_1 yz + U_0 z^2, \quad U_0, U_1, U_2 \in \mathrm{GF}(2)[x], \deg(U_i) < b$$

Evaluate at 5 points: $\{(0,1),\ (1,1),\ (x,1),\ (x+1,1),\ (1,0)\}$

$$\begin{pmatrix} u(0,1) \\ u(1,1) \\ u(x,1) \\ u(x+1,1) \\ u(1,0) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & x & x^2 \\ 1 & x+1 & x^2+1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U_0 \\ U_1 \\ U_2 \end{pmatrix} = \begin{pmatrix} U_0 \\ U_0 + U_1 + U_2 \\ U_0 + (x)U_1 + (x^2)U_2 \\ U_0 + (x+1)U_1 + (x^2+1)U_2 \\ U_2 \end{pmatrix}$$

A naïve implementation cost: $6 \times \texttt{add} + 2 \times \texttt{shift} + 2 \times \texttt{Smul}$.
First and last evaluations are trivial.

Evaluation is Matrix-vector multiplication

After splitting, operands are quadratic polynomials

$$u(y,z) = U_2 y^2 + U_1 yz + U_0 z^2, \quad U_0, U_1, U_2 \in \mathrm{GF}(2)[x], \deg(U_i) < b$$

Evaluate at 5 points: $\{(0,1),\ (1,1),\ (x,1),\ (x+1,1),\ (1,0)\}$

$$\begin{pmatrix} u(0,1) \\ u(1,1) \\ u(x,1) \\ u(x+1,1) \\ u(1,0) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & x & x^2 \\ 1 & x+1 & x^2+1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} U_0 \\ U_1 \\ U_2 \end{pmatrix} = \begin{pmatrix} U_0 \\ U_0 + U_1 + U_2 \\ U_0 + (x)U_1 + (x^2)U_2 \\ U_0 + (x+1)U_1 + (x^2+1)U_2 \\ U_2 \end{pmatrix}$$

A naïve implementation cost: $8 \times$ add $+ 4 \times$ shift.
First and last evaluations are trivial.

Search a sequence of operations on matrix lines

Start from the “empty” matrix, search a path to the goal

No temporaries: in-place operations.

$$\begin{pmatrix} l_{-1} & 1 & 0 & 0 \\ l_{-2} & 0 & 1 & 0 \\ l_{-3} & 0 & 0 & 1 \\ \hline l_1 & 0 & 0 & 0 \\ l_2 & 0 & 0 & 0 \\ l_3 & 0 & 0 & 0 \end{pmatrix} \overset{l_1 \leftarrow l_{-1}+l_{-2}}{\rightsquigarrow} \begin{pmatrix} l_{-1} & 1 & 0 & 0 \\ l_{-2} & 0 & 1 & 0 \\ l_{-3} & 0 & 0 & 1 \\ \hline l_1 & 1 & 1 & 0 \\ l_2 & 0 & 0 & 0 \\ l_3 & 0 & 0 & 0 \end{pmatrix}$$

$$l_1 \leftarrow (x) l_{-2} + (x^2) l_{-3} \quad \downarrow \qquad \ddots \qquad \vdots$$

$$\begin{pmatrix} l_{-1} & 1 & 0 & 0 \\ l_{-2} & 0 & 1 & 0 \\ l_{-3} & 0 & 0 & 1 \\ \hline l_1 & 0 & x & x^2 \\ l_2 & 0 & 0 & 0 \\ l_3 & 0 & 0 & 0 \end{pmatrix} \quad \cdots \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hline 1 & 1 & 1 \\ 1 & x & x^2 \\ 1 & x+1 & x^2+1 \end{pmatrix}$$

Order of nontrivial values doesn’t matter.

Paths with different costs

even with same number of steps

Here two partial paths are shown.

$$\left(\begin{array}{c:ccc} l_{-1} & 1 & 0 & 0 \\ l_{-2} & 0 & 1 & 0 \\ l_{-3} & 0 & 0 & 1 \\ \hline l_1 & 1 & 1 & 1 \\ l_2 & 0 & 0 & 0 \\ l_3 & 0 & 0 & 0 \end{array}\right) \underset{\rightsquigarrow}{l_2 \leftarrow l_{-2} + (x+1)\cdot l_{-3}} \left(\begin{array}{c:ccc} l_{-1} & 1 & 0 & 0 \\ l_{-2} & 0 & 1 & 0 \\ l_{-3} & 0 & 0 & 1 \\ \hline l_1 & 1 & 1 & 1 \\ l_2 & 0 & 1 & x+1 \\ l_3 & 0 & 0 & 0 \end{array}\right)$$

$$l_2 \leftarrow (x) l_{-2} + (x^2) l_{-3} \downarrow \qquad\qquad l_2 \leftarrow (x+1) l_2 + l_1 \downarrow$$

$$\left(\begin{array}{c:ccc} l_{-1} & 1 & 0 & 0 \\ l_{-2} & 0 & 1 & 0 \\ l_{-3} & 0 & 0 & 1 \\ \hline l_1 & 1 & 1 & 1 \\ l_2 & 0 & x & x^2 \\ l_3 & 0 & 0 & 0 \end{array}\right) \underset{\rightsquigarrow}{l_2 \leftarrow l_2 + l_1} \left(\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hline 1 & 1 & 1 \\ 1 & x+1 & x^2+1 \\ 0 & 0 & 0 \end{array}\right)$$

Initial and final matrices coincide, but the cost is different.

Optimal evaluation sequence

The power of recycling

Path on the graph...

$$\left(\begin{array}{c:ccc} l_{-1} & 1 & 0 & 0 \\ l_{-2} & 0 & 1 & 0 \\ l_{-3} & 0 & 0 & 1 \\ \hline l_1 & 0 & 0 & 0 \\ l_2 & 0 & 0 & 0 \\ l_3 & 0 & 0 & 0 \end{array}\right) \underset{l_3 \leftarrow (x) l_{-2} + (x^2) l_{-3}}{\overset{l_1 \leftarrow l_{-1} + l_{-2} + l_{-3}}{\rightsquigarrow}} \left(\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hline 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & x & x^2 \end{array}\right) \underset{l_3 \leftarrow l_3 + l_1}{\overset{l_2 \leftarrow l_3 + l_{-1}}{\rightsquigarrow}} \left(\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hline 1 & 1 & 1 \\ 1 & x & x^2 \\ 1 & x+1 & x^2+1 \end{array}\right)$$

Total cost: $5 \times \texttt{add} + 2 \times \texttt{shift}$

Naïve was: $8 \times \texttt{add} + 4 \times \texttt{shift}$

...immediately translates to temporary-less evaluation sequence

$$L_1 = U_0 + U_1 + U_2;\ L_3 = (x) \cdot U_2 + (x^2) \cdot U_3;$$
$$L_2 = L_3 + U_0;\ L_3 = L_3 + L_1$$

After recursive multiplication $w(\alpha,\beta) = u(\alpha,\beta)v(\alpha,\beta)$

$$\begin{pmatrix} w(0,1) \\ w(1,1) \\ w(x,1) \\ w(x+1,1) \\ w(1,0) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & x & x^2 & x^3 & x^4 \\ 1 & x+1 & x^2+1 & (x+1)^3 & x^4+1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} W_0 \\ W_1 \\ W_2 \\ W_3 \\ W_4 \end{pmatrix}$$

Graph search for interpolation too [ISSAC2007].
Cost found: $9 \times \texttt{add} + 1 \times \texttt{shift} + 1 \times \texttt{Smul} + 2 \times \texttt{Sdiv}$
Multiplication by x^3+1, exact divisions by $x+1, x^2+x$.

see

A Toom-3 in $\mathrm{GF}(2)[x]$ without divisions is not possible.

Final recomposition, doubly length coefficients

$$\begin{array}{l} \quad\quad [\dots W3 \dots][\dots W1 \dots] \quad\quad \oplus \\ [\dots W4 \dots][\dots W2 \dots][\dots W0 \dots] = w \end{array}$$

Thresholds for NTL-based implementations

Range where each algorithm is the fastest

Algorithm	operand degree (bits)			asymptotic
Naïve		$<$	190	$\mathrm{O}(d^2)$
Karatsuba	190	...	360	$\mathrm{O}(d^{\log_2 3})$
Toom-3	360	...	8,000	$\mathrm{O}(d^{\log_3 5})$
Toom-4	8,000	...	15,000	$\mathrm{O}(d^{\log_4 7})$
Schönhage-FFT	15,000	$<$		$\mathrm{O}(d \log d \log \log d)$

Those values highly depend on implementation, architecture...

Algorithms in blue where implemented by Paul Zimmermann

What else you can find on the paper?

Only about 10 pages of the paper reported in this presentation

Details skipped during presentation

- Heuristics for graph search.
- Operands with very different size
- Bivariate (and sketches on multivariate)
- Results for characteristic 0 ($\mathbb{Z}[x]$ and $\mathbb{Z}$, + squaring)

▸ Unbalanced

The title of the paper is much longer!

Towards Optimal Toom-Cook Multiplication for Univariate and Multivariate Polynomials in Characteristic 2 and 0

▸ Download

That's all !

Thank you very much for your kind attention

Questions?

Full paper too is available on web.

4 More on computations

- Exact division
- Unbalanced multiplication
- Choice of points

back to index

Exact division

detailed only for $D = x^n + 1 \in \mathrm{GF}(2)[x]$

We start from an element $\mathrm{GF}(2)[x] \ni a = qD$, whose degree is $\deg(a) = d + n$. We want the quotient q. Compute with $2^k n \leqslant d$.

$$q \equiv a \cdot (1 + x^n) \cdot (1 + x^{2n}) \cdots (1 + x^{2^k n}) \pmod{x^{d+1}}$$

Division can be performed limb by limb starting from less significant one, obtaining linear complexity.

Division limb by limb obtain linear complexity

for $i = 0 \ldots d/w$

$a_i \leftarrow a_i \cdot D^{-1} \pmod{x^w}$

$a_{i+1} \leftarrow a_{i+1} - \frac{a_i \cdot D}{x^w} = a_{i+1} - a_i >> (w - n)$

Thanks to Jörg Arndt for suggesting a clean description

Splitting for unbalanced operands

Toom-2.5

Degree 2 $\times$ degree 1 $\rightsquigarrow$ degree 3.

3 parts $\times$ 2 parts $\rightsquigarrow$ 4 parts.

$$\begin{array}{rclcccc}
\mathrm{GF}(2)[x] & & \rightsquigarrow & & & & \mathrm{GF}(2)[x]\,[y,z] \\
u = & [\ldots\ldots\ldots] & \rightsquigarrow & [\ldots]\cdot y^2 + & [\ldots]\cdot yz & +[\ldots]\cdot z^2 & = \mathfrak{u} \\
v = & [\ldots\ldots] & \rightsquigarrow & & [\ldots]\cdot y & +[\ldots]\cdot z & = \mathfrak{v}
\end{array}$$

Unbalanced Toom-3

back to balanced

Degree 3 $\times$ degree 1 $\rightsquigarrow$ degree 4.

4 parts $\times$ 2 parts $\rightsquigarrow$ 5 parts.

$$\begin{array}{rccccc}
\mathrm{GF}(2)[x] & \rightsquigarrow & & & & \mathrm{GF}(2)[x]\,[y,z] \\
[\ldots\ldots\ldots\ldots] & \rightsquigarrow & [\ldots]\cdot y^3 + [\ldots]\cdot y^2 z + & [\ldots]\cdot yz^2 & +[\ldots]\cdot z^3 & \\
[\ldots\ldots] & \rightsquigarrow & & [\ldots]\cdot y & +[\ldots]\cdot z &
\end{array}$$

How to choose evaluation/interpolation points

Points chosen for the results gives small degree increase and small cost for ES/IS. Different choices are possible.

An anonymous referee and Richard Brent suggested the use of x^w, x^w+1 for w-bits CPU. ES and IS basically remain the same.

When working on $\mathrm{GF}(2^n)[x]$ we are working on $\mathrm{GF}(2)[x]_{/p}[X]$, so we have to choose the use of $x, x+1$ or $X, X+1$, test for any particular implementation.