Frobenius Additive Fast Fourier Transform

Wen-Ding Li

Research Center for Information Technology Innovation, Academia Sinica, Taiwan

July 19, 2018

ISSAC 2018, New York, USA

Joint work with Ming-Shing Chen, Po-Chun Kuo, Chen-Mou Cheng, Bo-Yin Yang

Polynomial Multiplication over $\mathbb{F}_2$

- Schoolbook : $O(n^2)$
- Karatsuba or Toom-Cook : $O(n^\omega)$, $1 < \omega < 2$
- Fast Fourier Transform (FFT) : $\widetilde{O}(n)$

Multiplication with FFT

Fourier transform of $f \in \mathbb{F}[x]$: Evaluation of f in some zero set $Z \subset \mathbb{F}$.

Multiplication with FFT

Fourier transform of $f \in \mathbb{F}[x]$: Evaluation of f in some zero set $Z \subset \mathbb{F}$.

How do we multiply $h = f \cdot g$ in $\mathbb{F}[x]$?

Multiplication with FFT

Fourier transform of $f \in \mathbb{F}[x]$: Evaluation of f in some zero set $Z \subset \mathbb{F}$.

How do we multiply $h = f \cdot g$ in $\mathbb{F}[x]$?

- Evaluate f and g at points of some zero set $Z \subset \mathbb{F}$
- Multiply pointwise to obtain $\{f(\alpha) \cdot g(\alpha), \alpha \in Z\}$
- Interpolate: recover h from $\{f(\alpha) \cdot g(\alpha), \alpha \in Z\}$

Multiplication with FFT

Fourier transform of $f \in \mathbb{F}[x]$: Evaluation of f in some zero set $Z \subset \mathbb{F}$.

How do we multiply $h = f \cdot g$ in $\mathbb{F}[x]$?

- Evaluate f and g at points of some zero set $Z \subset \mathbb{F}$
- Multiply pointwise to obtain $\{f(\alpha) \cdot g(\alpha), \alpha \in Z\}$
- Interpolate: recover h from $\{f(\alpha) \cdot g(\alpha), \alpha \in Z\}$

Multiplication in $\mathbb{F}_2[x]$

- Not many evaluation points in $\mathbb{F}_2 \Rightarrow$ **work in an extension field**
- Naive method: $\mathbb{F}_2[x] \rightsquigarrow \mathbb{F}_{2^d}[x]$

Multiplication with FFT

Fourier transform of $f \in \mathbb{F}[x]$: Evaluation of f in some zero set $Z \subset \mathbb{F}$.

How do we multiply $h = f \cdot g$ in $\mathbb{F}[x]$?

- Evaluate f and g at points of some zero set $Z \subset \mathbb{F}$
- Multiply pointwise to obtain $\{f(\alpha) \cdot g(\alpha), \alpha \in Z\}$
- Interpolate: recover h from $\{f(\alpha) \cdot g(\alpha), \alpha \in Z\}$

Multiplication in $\mathbb{F}_2[x]$

- Not many evaluation points in $\mathbb{F}_2$ $\Rightarrow$ **work in an extension field**
- Naive method: $\mathbb{F}_2[x] \rightsquigarrow \mathbb{F}_{2^d}[x] \Rightarrow$ incurs d-times penalty.

The Kronecker segmentation

- Schönhage's ternary FFT (GF2x: Brent, Gaudry, Thome, Zimmermann)
  $\mathbb{F}_2[x] \rightsquigarrow \mathbb{F}_2[x]_{<M}[y] \rightsquigarrow \mathbb{F}_2[x]/(x^{2L}+x^L+1)[y]$, $y = x^M$, $L >= M$
- Mixed Radix FFT over $\mathbb{F}_{2^{60}}$ (ISSAC 2016: Harvey, van der Hoeven, Lecerf)
  $\mathbb{F}_2[x] \rightsquigarrow \mathbb{F}_2[x]_{<30}[y] \rightsquigarrow \mathbb{F}_{2^{60}}[y]$, $y = x^{30}$
- Additive FFT over $\mathbb{F}_{2^{256}}$ (Chen, Cheng, Kuo, Li, Yang - 2017)
  $\mathbb{F}_2[x] \rightsquigarrow \mathbb{F}_2[x]_{<128}[y] \rightsquigarrow \mathbb{F}_{2^{256}}[y]$, $y = x^{128}$

Pack half as many bits in each coefficients as the extension field

The Kronecker segmentation

- Schönhage's ternary FFT (GF2x: Brent, Gaudry, Thome, Zimmermann)
  $\mathbb{F}_2[x] \rightsquigarrow \mathbb{F}_2[x]_{<M}[y] \rightsquigarrow \mathbb{F}_2[x]/(x^{2L}+x^L+1)[y]$, $y = x^M$, $L >= M$
- Mixed Radix FFT over $\mathbb{F}_{2^{60}}$ (ISSAC 2016: Harvey, van der Hoeven, Lecerf)
  $\mathbb{F}_2[x] \rightsquigarrow \mathbb{F}_2[x]_{<30}[y] \rightsquigarrow \mathbb{F}_{2^{60}}[y]$, $y = x^{30}$
- Additive FFT over $\mathbb{F}_{2^{256}}$ (Chen, Cheng, Kuo, Li, Yang - 2017)
  $\mathbb{F}_2[x] \rightsquigarrow \mathbb{F}_2[x]_{<128}[y] \rightsquigarrow \mathbb{F}_{2^{256}}[y]$, $y = x^{128}$

Pack half as many bits in each coefficients as the extension field

Factor-of-two loss!

The Frobenius Fourier transform - ISSAC 2017

Directly compute Fourier transform of a polynomial f in $\mathbb{F}_2[x]_{<n}$:

$$\{f(1), f(\omega), f(\omega^2), \dots, f(\omega^{n-1})\}$$

where $\omega \in \mathbb{F}_{2^d}$ primitive root of unity.

The Frobenius Fourier transform - ISSAC 2017

Directly compute Fourier transform of a polynomial f in $\mathbb{F}_2[x]_{<n}$:

$$\{f(1), f(\omega), f(\omega^2), \dots, f(\omega^{n-1})\}$$

where $\omega \in \mathbb{F}_{2^d}$ primitive root of unity.

Save some computation by using the Frobenius automorphism:

$$f(w^2) = f(\phi(w)) = \phi(f(w)) = (f(w))^2$$

$\Rightarrow$ For each orbit w, $\phi(w)$, $\phi^{\circ 2}(w)$, $\phi^{\circ 3}(w)$, ..., we only need to compute at one point: $f(w)$ and all other values $\phi^{\circ 2}(f(w))$, $\phi^{\circ 3}(f(w))$, ... are determined.

The Frobenius Fourier transform - ISSAC 2017

Directly compute Fourier transform of a polynomial f in $\mathbb{F}_2[x]_{<n}$:

$$\{f(1), f(\omega), f(\omega^2), \ldots, f(\omega^{n-1})\}$$

where $\omega \in \mathbb{F}_{2^d}$ primitive root of unity.

Save some computation by using the Frobenius automorphism:

$$f(w^2) = f(\phi(w)) = \phi(f(w)) = (f(w))^2$$

$\Rightarrow$ For each orbit w, $\phi(w)$, $\phi^{\circ 2}(w)$, $\phi^{\circ 3}(w)$, ..., we only need to compute at one point: $f(w)$ and all other values $\phi^{\circ 2}(f(w))$, $\phi^{\circ 3}(f(w))$, ... are determined.

Result: d-times faster than naive method.

Cantor's FFT and its derivatives

- Cantor give an "analogue of the fast Fourier transform" which efficiently evaluates a polynomial on some additive subgroup Z of $\mathbb{F}_{p^{p^k}}$ in $O(n(\log n)^2)$ time for $n = |Z|$.
    - Based on a tower $\mathbb{F}_p, \mathbb{F}_{p^p}, \mathbb{F}_{p^{p^2}}, \dots$ of Artin-Schreier extensions of $\mathbb{F}_p$
- Gao and Mateer improved it to $O(n \log n \log(\log n))$ when $p = 2$ and $f \in \mathbb{F}_{2^{2^k}}[x]$
- We showed that van der Hoeven and Larrieu's idea of using Frobenius automorphism to accelerate polynomial multiplication beautifully generalizes to Cantor-Gao-Mateer-FFT

Additive FFT

Let $s(x) = x^2 + x$, $s_0(x) = x$ and

$$s_i(x) := \underbrace{s(s(\cdots s}_{i \text{ times}}(x)\cdots)) = s^{\circ i}(x)$$

- Let W_i be the zero set of $s_i(x) = \prod_{\omega \in W_i}(x - \omega)$, then

$$\mathbb{F}_2 = W_1 \subset W_2 \subset \cdots \subset \widetilde{\mathbb{F}_2}$$

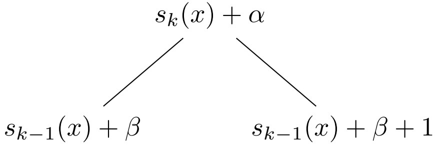
- Since s_i's are linear, W_i's are vector spaces over $\mathbb{F}_2$
- Since $s_{2^k} = x^{2^{2^k}} + x$, W_{2^k} is a field $\mathbb{F}_{2^{2^k}}$.
  e.g. $W_1 = \mathbb{F}_2$, $W_2 = \mathbb{F}_{2^2}$, $W_4 = \mathbb{F}_{2^4}$, $W_8 = \mathbb{F}_{2^8}$,...
- Cantor showed that there is a basis $(v_0, v_1, v_2, \ldots,)$ such that $W_i =$ span$\{v_0, v_1, \ldots, v_{i-1}\}$ and $s(v_i) = v_i^2 + v_i = v_{i-1}$
- We'll denote $a_0 v_0 + a_1 v_1 + \ldots + a_{d-1} v_d$ as $a_{d-1} a_{d-1} \ldots a_0$.
  e.g. 1101 is $v_3 + v_2 + v_0$.

Additive FFT - Subproduct Tree

$s_k(x) + \alpha$ can be written as the product of

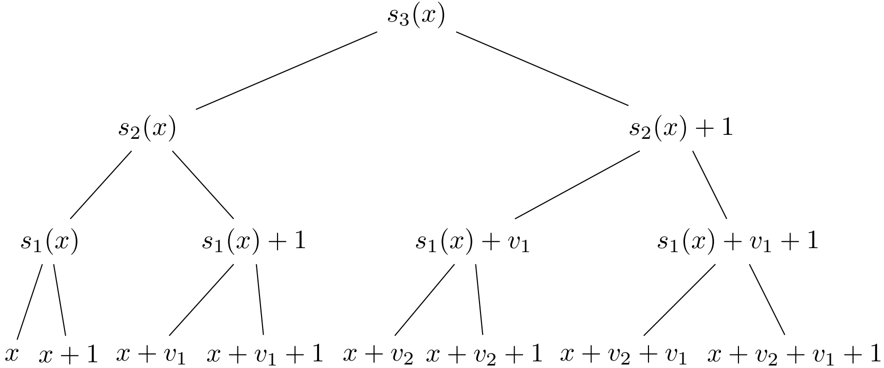
$$s_{k-1}(x) + \beta \text{ and } s_{k-1}(x) + \beta + 1,$$

where $\beta^2 + \beta = \alpha$.



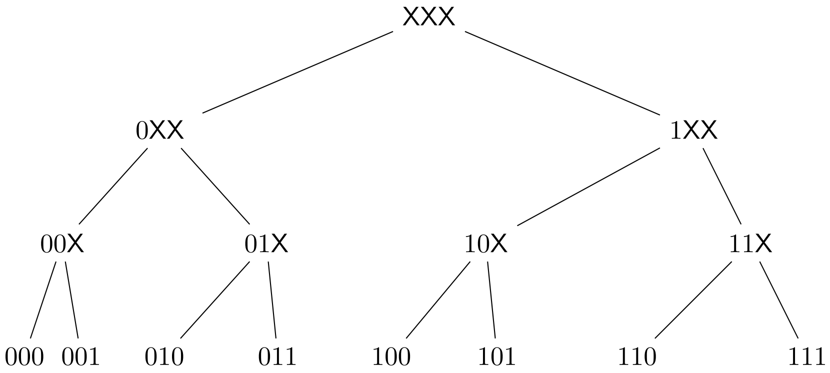
right child $=$ left child $+1$

Additive FFT

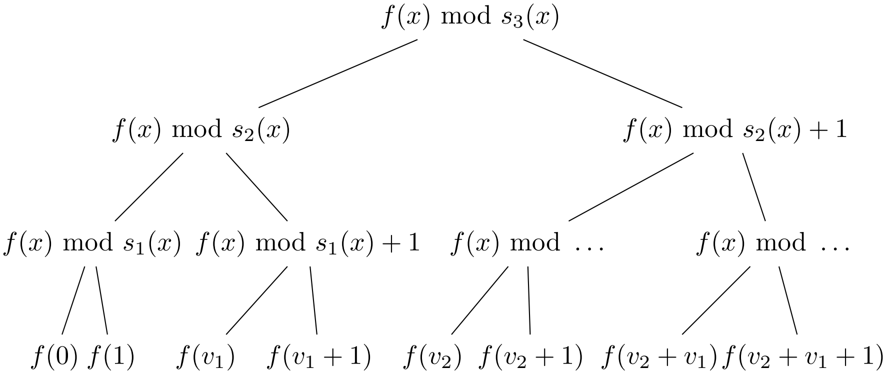


Additive FFT

The roots of polynomial in subproduct tree. The “X” means it could take 0 or 1.



Additive FFT



Additive FFT

Let $\big(f(x) \bmod s_n(x) + \alpha\big) = P(x)s_{n-1}(x) + Q(x)$ [Gao-Mateer], then

$$f(x) \mod s_n(x) + \alpha$$

$f(x) \bmod s_{n-1}(x) + \beta$
$= Q(x) + \beta P(x)$

$f(x) \bmod s_{n-1}(x) + \beta + 1$
$= Q(x) + \beta P(x) + P(x)$

Let the left child be $f_0(x)$ and the right child be $f_1(x)$, then

$$f_0(x) = Q(x) + \beta P(x)$$
$$f_1(x) = P(x) + f_0(x)$$

By applying this recursively, we get

$$\{f(x) \bmod x + \omega | s_n(\omega) = \alpha\} = \{f(\omega) | \omega \in W_i + \gamma\}$$

where $s_n(\gamma) = \alpha$

Frobenius Additive FFT

Question: Given d a power of two, when computing **additive** FFT of f in $\mathbb{F}_{2^d}[x]$, can we achieve d-times speedup if f actually admits only coefficients in $\mathbb{F}_2$?

Frobenius Additive FFT

Question: Given d a power of two, when computing **additive** FFT of f in $\mathbb{F}_{2^d}[x]$, can we achieve d-times speedup if f actually admits only coefficients in $\mathbb{F}_2$?

Save some computation by using the Frobenius automorphism:

$$f(w^2) = (f(w))^2$$

Frobenius Additive FFT

Question: Given d a power of two, when computing **additive** FFT of f in $\mathbb{F}_{2^d}[x]$, can we achieve d-times speedup if f actually admits only coefficients in $\mathbb{F}_2$?

Save some computation by using the Frobenius automorphism:

$$f(w^2) = (f(w))^2$$

$\Rightarrow$ **If we have $f(w)$, $f(w^2)$ can be obtained efficiently. Only need to evaluate a subset of the original points**

Orbits under the action of $\phi : x \mapsto x^2$

Denote the Orbit of w under the action ϕ be

$$\begin{aligned} \mathrm{Orb}_w &= \{w, \phi(w), \phi^{\circ 2}(w), \phi^{\circ 3}(w), \phi^{\circ 4}(w), \ldots\} \\ &= \{w, w^2, w^4, w^8, w^{16}, \ldots\} \end{aligned}$$

- For $w \in W_{i+1} \setminus W_i$, $|\mathrm{Orb}_w| = 2^{\lfloor \lg i \rfloor + 1}$
- How the action affect the points:

$$\phi^{\circ 2^k}(x) = s_{2^k}(x) + x$$

Change the position whose distance is 2^k from most significant bits

Main Result: the Cross section of the orbit

Let $\Sigma_0 = \{0\}$, and $\forall k > 0$, let

$$\Sigma_k = \left\{ v_{k-1} + j_1 v_{k-2} + \cdots + j_{k-1} v_0 : \begin{array}{l} j_i = 0 \text{ if } i \text{ is a power of 2,} \\ j_i \in \{0,1\} \text{ otherwise.} \end{array} \right\}$$
$$= 100\text{X}0\text{XXX}0\text{XXXXXXX}0\text{XX}\ldots$$

Theorem

Σ_k is a cross section of $W_k \setminus W_{k-1}$. That is, $\forall k > 0$, $\forall w \in W_k \setminus W_{k-1}$, there exists exactly one $\sigma \in \Sigma_k$ such that $\phi^{\circ j}(\sigma) = w$ for some j.

Main Result: the Cross section of the orbit

Let $\Sigma_0 = \{0\}$, and $\forall k > 0$, let

$$\Sigma_k = \left\{ v_{k-1} + j_1 v_{k-2} + \cdots + j_{k-1} v_0 : \begin{array}{l} j_i = 0 \text{ if } i \text{ is a power of 2,} \\ j_i \in \{0,1\} \text{ otherwise.} \end{array} \right\}$$

$$= 100\text{X}0\text{XXX}0\text{XXXXXXX}0\text{XX}\ldots$$

Theorem

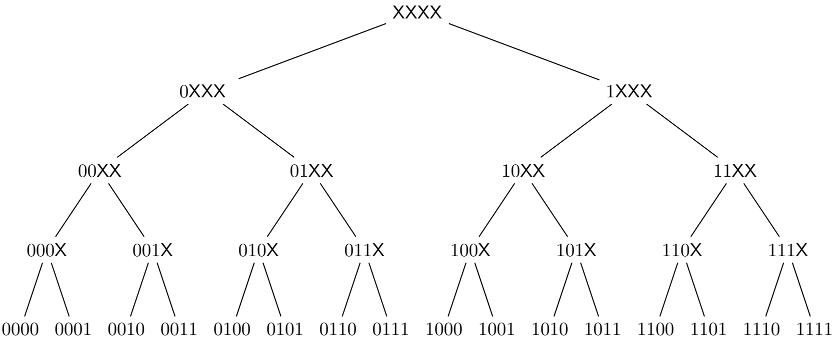
Σ_k is a cross section of $W_k \setminus W_{k-1}$. That is, $\forall k > 0$, $\forall w \in W_k \setminus W_{k-1}$, there exists exactly one $\sigma \in \Sigma_k$ such that $\phi^{\circ j}(\sigma) = w$ for some j.

A cross section of W_m is

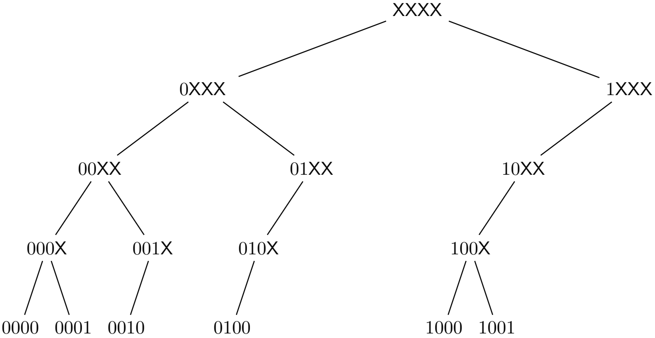
$$\Sigma_0 \cup \Sigma_1 \cup \Sigma_2 \cup \ldots \cup \Sigma_m \quad .$$

.

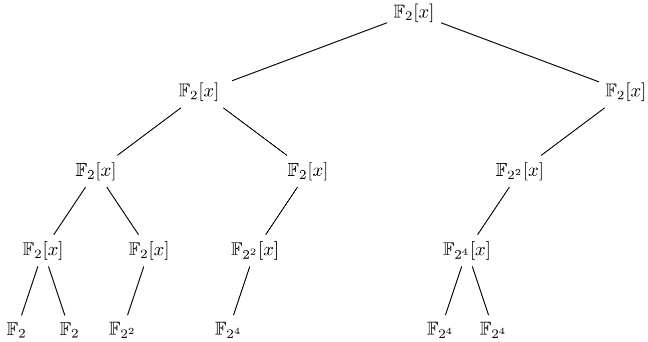
Truncated additive FFT



Truncated additive FFT



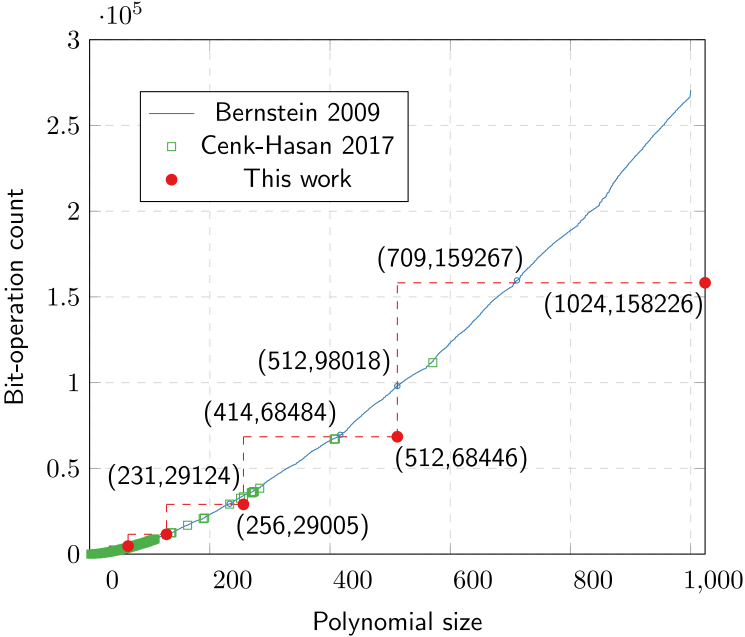
Truncated additive FFT



New speed records in terms of bit-operation count

- Consider subfield properties to accelerate the constant multiplications in FFT
  - Use tower field representations
- Use common subexpression elimination techniques to further reduce the number of bit operations

New speed records in terms of bit-operation count



New speed records on modern CPUs

We need to use the PCLMULQDQ instruction to multiply in $\mathbb{F}_{2^{128}}[x]$:
Cross-section of size 2^{m-7} we use to enable truncated additive FFT:

$$\begin{aligned} \Sigma &= \{v_k + j_{64}v_{k-64} + j_{65}v_{k-65} + \cdots + j_{2^{m-7}-1}v_{k-63-2^{m-7}} : j_i\} \\ &= \{1\overbrace{00\cdots0}^{64}\overbrace{XX\cdots X}^{m-7}\} \end{aligned}$$

Table: Timing of multiplications in $\mathbb{F}_2[x]_{<n}$ on Intel Skylake Xeon E3-1275 v5 @ 3.60GHz (10^{-3} sec.)

$\log_2(n/64)$	16	17	18	19	20	21	22	23
This work, $\mathbb{F}_{2^{128}}$	9	20	41	88	192	418	889	1865
FDFT [c]	11	24	56	127	239	574	958	2465
ADFT	16	34	74	175	382	817	1734	3666
FFT over $\mathbb{F}_{2^{60}}$ [b]	22	51	116	217	533	885	2286	5301
gf2x [a]	23	51	111	250	507	1182	2614	6195

[a] Version 1.2. Available from http://gf2x.gforge.inria.fr/
[b] SVN r10663. Available from svn://scm.gforge.inria.fr/svn/mmx
[c] SVN r10681. Available from svn://scm.gforge.inria.fr/svn/mmx