fflonk Specification

@saitima

2024

1 Introduction

fflonk is an optimized variant of the Plonk [?]. This document outlines our implementation of the fflonk [?] protocol. The primary motivation behind fflonk is to reduce the verifier's complexity while maintaining the security guarantees of the original Plonk protocol. Our approach involves efficient polynomial combination and batch opening techniques.

2 Preliminaries

We assume familiarity with the Plonk protocol [?] and its underlying concepts. Let $\mathbb{F}$ be a prime field, and ω be a primitive N-th root of unity in $\mathbb{F}$, where N is the trace length.

3 Polynomial Definitions and Degrees

The protocol uses the following polynomials:

- Selector polynomials(canonical order): $q_A, q_B, q_C, q_M, q_{Const} \in \mathbb{F}[X]_{\leq N-1}$
- Permutation polynomials: $S_{\sigma 1}, S_{\sigma 2}, S_{\sigma 3} \in \mathbb{F}[X]_{\leq N-1}$
- Witness polynomials(canonical order): $a, b, c \in \mathbb{F}[X]_{\leq N-1}$
- Main gate quotient: $t_{\text{main}} \in \mathbb{F}[X]_{\leq 2N-3}$
- Grand product: $Z \in \mathbb{F}[X]_{\leq N-1}$
- Copy-permutation quotients: $t_1 \in \mathbb{F}[X]_{\leq 3N-4}$, $t_2 \in \mathbb{F}[X]_{\leq N-1}$
- Combined polynomials: $C_0 \in \mathbb{F}[X]_{\leq 8N-1}$, $C_1 \in \mathbb{F}[X]_{\leq 8N-9}$, $C_2 \in \mathbb{F}[X]_{\leq 9N-10}$
- Opening polynomials: $W \in \mathbb{F}[X]_{\leq 9N-28}$, $W' \in \mathbb{F}[X]_{\leq 9N-11}$

4 Protocol Specification

4.1 Polynomial Identities

4.1.1 Main Gate Identity

$$q_M(X)\cdot a(X)\cdot b(X)+q_A(X)\cdot a(X)+q_B(X)\cdot b(X)+q_C(X)\cdot c(X)+q_{Const}(X)+PI(X)\equiv 0 \pmod{X^N-1} \tag{1}$$

where $PI(X)$ is the public input polynomial.

4.1.2 First Copy-Permutation Identity

$$\begin{aligned} &Z(X)\cdot(a(X)+\beta X+\gamma)\cdot(b(X)+\beta k_1 X+\gamma)\cdot(c(X)+\beta k_2 X+\gamma)- \\ &Z(X\omega)\cdot(a(X)+\beta S_{\sigma 1}(X)+\gamma)\cdot(b(X)+\beta S_{\sigma 2}(X)+\gamma)\cdot(c(X)+\beta S_{\sigma 3}(X)+\gamma)\equiv 0 \pmod{X^N-1} \end{aligned} \tag{2}$$

where β, γ are random challenges, and k_1, k_2 are quadratic non-residues.

4.1.3 Second Copy-Permutation Identity

$$(Z(X)-1)\cdot L_1(X)\equiv 0 \pmod{X^N-1} \tag{3}$$

where $L_1(X)$ is the first Lagrange basis polynomial.

4.2 Prover Rounds

4.2.1 Round 1: Preprocessing polynomials

Combined polynomial:

$$C_0(X)=q_A(X^8)+Xq_B(X^8)+X^2q_C(X^8)+X^3q_M(X^8)+X^4q_{Const}(X^8)+X^5S_{\sigma 1}(X^8)+X^6S_{\sigma 2}(X^8)+X^7S_{\sigma 3}(\tag{4}$$

4.2.2 Round 2: Witness polynomials and main gate identity

Combined polynomial:

$$C_1(X)=a(X^4)+Xb(X^4)+X^2c(X^4)+X^3t_{\text{main}}(X^4) \tag{5}$$

4.2.3 Round 3: Copy-Permutation

Combined polynomial:

$$C_2(X)=Z(X^3)+Xt_1(X^3)+X^2t_2(X^3) \tag{6}$$

4.3 Polynomial Openings

The prover provides evaluations of all polynomials (including preprocessing polynomials) in the same order as the polynomials are defined:

- Preprocessing polynomials: at points $\{h_0, w_0 h_0, \ldots, w_0^7 h_0\}$
- Round 1 polynomials: at points $\{h_1, w_1 h_1, \ldots, w_1^3 h_1\}$
- Round 2 polynomials: at points $\{h_2, w_2 h_2, \ldots, w_2^3 h_2\}$ and $\{\omega^{1/3} h_2, \omega^{1/3} w_2 h_2, \ldots, \omega^{1/3} w_2^3 h_2\}$

where w_i is a k_i-th root of unity, $h_i = r^{\text{LCM}(8,4,3)/k_i}$, and r is verifier's random point.

4.4 Opening

1. Prover computes and sends commitment to $W(X)$:

$$W(X) = \sum_i \frac{\alpha^i \cdot Z_{T \setminus S_i}(X) \cdot (C_i(X) - r_i(X))}{Z_T(X)} \tag{7}$$

where α is a random challenge from the verifier and $r_i(X)$ are the opening polynomials.

2. Verifier sends random challenge y.
3. Prover computes and sends commitment to $W'(X)$:

$$W'(X) = \frac{W(X) \cdot Z_T(y) - \sum_i \alpha^i \cdot (C_i(X) - r_i(y)) \cdot Z_{T \setminus S_i}(X)}{X - y} \tag{8}$$

Note: The verifier can construct each $r_i(y)$ from existing evaluations using barycentric interpolation(Appendix A).

4. Verifier checks the pairing equation:

$$e([C(x)] - [r(y) \cdot G_1] - [Z_T(y)/Z_{T \setminus S_0}(y) \cdot W] + [y \cdot W'], G_2) \cdot e(-W', x \cdot G_2) = 1 \tag{9}$$

where

$$C(x) = C_0(x) + \alpha \cdot \frac{Z_{T \setminus S_1}(y)}{Z_{T \setminus S_0}(y)} \cdot C_1(x) + \alpha^2 \cdot \frac{Z_{T \setminus S_2}(y)}{Z_{T \setminus S_0}(y)} \cdot C_2(x)$$

and

$$r(y) = r_0(y) + \alpha \cdot \frac{Z_{T \setminus S_1}(y)}{Z_{T \setminus S_0}(y)} \cdot r_1(y) + \alpha^2 \cdot \frac{Z_{T \setminus S_2}(y)}{Z_{T \setminus S_0}(y)} \cdot r_2(y)$$

5 Conclusion

This document is a preliminary draft of our protocol implementation and is subject to updates and revisions.

6 Appendix

6.1 Barycentric Interpolation

Barycentric interpolation is used in our fflonk implementation to efficiently reconstruct polynomial evaluations at arbitrary points. This method is particularly useful for the verifier to compute evaluations of the combined polynomials $C_i(X)$ at the challenge point y without having access to the full polynomials.

6.1.1 Lagrange Basis Polynomials

For preprocessing (8 polynomials) and first round (4 polynomials):

$$L_i(X) = \frac{w_i}{N \cdot h^{N-1}} \cdot \frac{X^N - h^N}{X - w_i h} \tag{10}$$

For the copy-permutation round (3 polynomials), which requires openings at shifted points:

For points in the original interpolation set:

$$L_i(X) = \frac{w_i}{N \cdot (h^{2N-1} - (w^{1/N}h)^N h^{N-1})} \cdot \frac{X^{2N} - X^N(h^N + (w^{1/N}h)^N) + (hw^{1/N}h)^N}{X - hw_i} \tag{11}$$

For points in the shifted set:

$$L_j(X) = \frac{w_j}{N \cdot (w^{1/N}h)^{2N-1} - h^N(w^{1/N}h)^{N-1}} \cdot \frac{X^{2N} - X^N(h^N + (w^{1/N}h)^N) + (hw^{1/N}h)^N}{X - w^{1/N}hw_j} \tag{12}$$

where N is the size of the interpolation set, h is the base point, w is the primitive root of the trace domain, and w_i, w_j are the N-th roots of unity.

6.1.2 Using Lagrange Basis Polynomials

The Lagrange basis polynomials are used to construct evaluations of the r_i polynomials at the point y provided by the verifier. The process is as follows:

For preprocessing polynomials (Round 0): The verifier constructs $r_0(y)$ using:

$$r_0(y) = C_0(h_0)L_{0,0}(y) + C_0(w_0h_0)L_{0,1}(y) + \cdots + C_0(w_0^7h_0)L_{0,7}(y) \tag{13}$$

where $L_{0,i}(y)$ are the Lagrange basis polynomials for the preprocessing round.

For first round polynomials (Round 1): The verifier constructs $r_1(y)$ using:

$$r_1(y) = C_1(h_1)L_{1,0}(y) + C_1(w_1h_1)L_{1,1}(y) + C_1(w_1^2h_1)L_{1,2}(y) + C_1(w_1^3h_1)L_{1,3}(y) \tag{14}$$

where $L_{1,i}(y)$ are the Lagrange basis polynomials for the first round.

For second round polynomials (Round 2): The verifier constructs $r_2(y)$ using:

$$\begin{aligned} r_2(y) =& C_2(h_2)L_{2,0}(y) + C_2(w_2h_2)L_{2,1}(y) + C_2(w_2^2h_2)L_{2,2}(y) + \\ & C_2(\omega^{1/3}h_2)L_{2,3}(y) + C_2(\omega^{1/3}w_2h_2)L_{2,4}(y) + C_2(\omega^{1/3}w_2^2h_2)L_{2,5}(y) \end{aligned} \tag{15}$$

where $L_{2,i}(y)$ are the Lagrange basis polynomials for the second round, including both the original and shifted interpolation sets.

6.2 Set Difference Polynomials

For each round i, we define:

$$Z_{T \setminus S_i}(X) = \frac{Z_T(X)}{Z_{S_i}(X)} \tag{16}$$

where:

- T is the union of all evaluation point sets across all rounds

- S_i is the set of evaluation points for round i
- $Z_T(X) = \prod_{\alpha \in T}(X - \alpha)$
- $Z_{S_i}(X) = \prod_{\alpha \in S_i}(X - \alpha)$

The degree of $Z_{T \setminus S_i}(X)$ is $|T| - |S_i|$.

6.2.1 Instantiations

In our specific implementation, the set difference polynomials are instantiated as follows:

$$Z_{T \setminus S_0}(X) = (X^4 - z) \cdot (X^3 - z) \cdot (X^3 - z\omega) \text{ where } Z_{T \setminus S_0} \in \mathbb{F}[X]_{\leq 10} \tag{17}$$

$$Z_{T \setminus S_1}(X) = (X^8 - z) \cdot (X^3 - z) \cdot (X^3 - z\omega) \text{ where } Z_{T \setminus S_1} \in \mathbb{F}[X]_{\leq 14} \tag{18}$$

$$Z_{T \setminus S_2}(X) = (X^8 - z) \cdot (X^4 - z) \text{ where } Z_{T \setminus S_2} \in \mathbb{F}[X]_{\leq 12} \tag{19}$$

so that

$$Z_T(X) = Z_{T \setminus S_0}(X) \cdot Z_{T \setminus S_1}(X) \cdot Z_{T \setminus S_2}(X) \tag{20}$$

$$Z_T(X) = (X^8 - z) \cdot (X^4 - z)(X^3 - z) \cdot (X^3 - z\omega) \text{ where } Z_T \in \mathbb{F}[X]_{\leq 18} \tag{21}$$

7 Tonelli-Shanks Algorithm for Cubic Roots

The Tonelli-Shanks algorithm, adapted for cubic roots, provides an efficient method for computing cubic roots in finite fields where they exist. This is particularly useful in the second round.

7.1 Algorithm

Let $\mathbb{F}_p$ be a finite field of prime order p, where $p \equiv 1 \pmod 3$. For $a \in \mathbb{F}_p$, we want to find x such that $x^3 \equiv a \pmod p$, if such an x exists.

1. Check if a is a cubic residue: if $a^{(p-1)/3} \not\equiv 1 \pmod p$, return none.
2. Write $p - 1 = 3^s \cdot q$ where $3 \nmid q$.
3. Find a cubic non-residue z: $z^{(p-1)/3} \not\equiv 1 \pmod p$.
4. Set $m = s$, $c = z^q \pmod p$, $t = a^q \pmod p$, $r = a^{(q+1)/3} \pmod p$.
5. While $t \not\equiv 1 \pmod p$:
   - Find the least i, $0 < i < m$, such that $t^{3^i} \equiv 1 \pmod p$.
   - If no such i exists, return null.
   - Update: $m = i$, $c = b^3 \pmod p$, $t = t \cdot c \pmod p$, $r = r \cdot b \pmod p$.
6. Return r.