# Plonk Arithmetization

*January 09, 2025*


## Distributed Lab

zkdl-camp.github.io

github.com/ZKDL-Camp

# Plan

# Introduction. PlonK: Five Ws

# What is PlonK?

PlonK is a type of zkSNARK:

- Groth16
- Halo2
- Marlin
- **PlonK**
- ...

# Who and When invented Plonk?

Ariel Gabizon, Zachary Williamson, Oana Ciobotaru introduced paper "PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge" in 2019

**PlonK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge**

Ariel Gabizon* — Aztec
Zachary J. Williamson — Aztec
Oana Ciobotaru

February 23, 2024

**Figure:** PlonK Paper. Date in Paper reflects the last update :)

# Why use Plonk?

**Focus on what you want:**

- ZKP for different tasks?
- Efficient proving times?
- Small-medium proof sizes?
- Flexibility?

# Where Plonk is used?

**zkVMs love Plonk!**

- Aztek Protocol (Noir)
- zkSync
- Dusk Network
- Mina Protocol

# PlonK Arithmetization

# Arithmetization

**Goal:** Write some program (computation) into math processing-prone form.

*Example*

**Public Input**: $x \in \mathbb{F}$
**Private Input**: $e \in \mathbb{F}$
**Output**: $e \times x + x - 1$

Let's split our program into the sequence of gates with left, right operands and output - circuit.

*Example*

We need **three gates** to encode our program:

1. **Gate #1**: left $e$, right $x$, output $u = e \times x$
2. **Gate #2**: left $u$, right $x$, output $v = u + x$
3. **Gate #3**: left $v$, right $x$, output $w = v + (-1)$

# Execution Trace

Then, form *execution trace table* — a matrix $T$ with columns $L$, $R$ and $O$.

## *Example*

| A | B | C |
|---|---|---|
| 2 | 3 | 6 |
| 6 | 3 | 9 |
| 9 | ✗ | 8 |

## *Remark*

Notice how the last row has no value in $B$ column (marked by ✗) — this is reasoned by the fact it is not a variable, but rather a constant, meaning it doesn't depend on execution.

# Encode the Program

Suppose you were given random matrix $T$. How could you tell if it is suitable for your circuit?

## *Solution*

Encode the circuit. Check $T$ using encoding:

1. **Gates** (gate constraints) — using matrix $Q$.
2. **Wires** (copy constraints) — using matrix $V$.

## *Definition (Gate Matrix)*

$Q$ matrix has one row per each gate with columns $Q_L$, $Q_R$, $Q_O$, $Q_M$, $Q_C$. If columns $A$, $B$ and $C$ of the execution trace table form valid evaluation of the circuit,

$$A_i Q_{Li} + B_i Q_{Ri} + A_i B_i Q_{Mi} + C_i Q_{Oi} + Q_{ci} = 0$$

# $Q$ Matrix

## Example

For our program, we would have a following $Q$ table:

| $Q_L$ | $Q_R$ | $Q_M$ | $Q_O$ | $Q_c$ |
|---|---|---|---|---|
| $0$ | $0$ | $1$ | $-1$ | $0$ |
| $1$ | $1$ | $0$ | $-1$ | $0$ |
| $1$ | $0$ | $0$ | $-1$ | $-1$ |

You can verify that our claim holds for aforementioned trace matrix:

$$2 \times 0 + 3 \times 0 + 2 \times 3 \times 1 + 6 \times (-1) + 0 = 0$$
$$6 \times 1 + 3 \times 1 + 6 \times 3 \times 0 + 9 \times (-1) + 0 = 0$$
$$9 \times 1 + 0 \times 0 + 9 \times 0 \times 0 + 8 \times (-1) + (-1) = 0$$

# $V$ Matrix

**Definition**

$V$ consists of indices of all inputs and intermediate values, so that if $T$ is a valid trace,

$$\forall i,j,k,l : (V_{i,j} = V_{k,l}) \Rightarrow (T_{i,j} = T_{k,l})$$

**Example**

For our program, $V$ would look like following:

| L | R | O |
|---|---|---|
| 0 | 1 | 2 |
| 2 | 1 | 3 |
| 3 | | 4 |

Here 0 is an index of $e$, 1 is an index of $x$, 2 — $u$, 3 — $v$ and 4 — output $w$.

# Custom Gates

Default Plonk: **addition** and **multiplication** gates. How to make it more *interesting*?

## *Solution*

$Q$ with it's 5 columns already allows for custom gates, however it is possible to include out own columns.

## *Example*

Our entire program may be encoded as one custom gate.

$Q$:

| $Q_L$ | $Q_R$ | $Q_M$ | $Q_o$ | $Q_c$ |
|---|---|---|---|---|
| 0 | 1 | 1 | $-1$ | $-1$ |

$V$:

| L | R | O |
|---|---|---|
| 0 | 1 | 2 |

$T$:

| A | B | C |
|---|---|---|
| 2 | 3 | 8 |

$$2 \times 0 + 3 \times 1 + 2 \times 3 \times 1 + 8 \times (-1) + (-1) = 0$$

# Public Inputs

Also need to encode public inputs.

**Idea**

Consider, as if we added new *selector* rows to $Q$ and tied them in $V$ and $T$.

**Example**

$Q$:

| $Q_L$ | $Q_R$ | $Q_M$ | $Q_o$ | $Q_c$ |
|---|---|---|---|---|
| -1 | 0 | 0 | 0 | 3 |
| -1 | 0 | 0 | 0 | 8 |
| 1 | 1 | 1 | -1 | 1 |
| 1 | -1 | 0 | 0 | 0 |

$V$:

| L | R | O |
|---|---|---|
| 0 | | |
| 1 | | |
| 2 | 0 | 3 |
| 1 | 3 | |

$T$:

| A | B | C |
|---|---|---|
| 3 | | |
| 8 | | |
| 2 | 3 | 8 |
| 8 | 8 | |

Now $Q$ and $V$ are not independent of evaluations.

**Solution**

We introduce another one-column matrix named $\Pi$ (public inputs).

# Wrap-Up

## *Example*

With only $Q$ modified, we have:

| $\Pi$ |
|---|
| 3 |
| 8 |
| 0 |
| 0 |

$Q$:

| $Q_L$ | $Q_R$ | $Q_M$ | $Q_o$ | $Q_c$ |
|---|---|---|---|---|
| -1 | 0 | 0 | 0 | 0 |
| -1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | -1 | 1 |
| 1 | -1 | 0 | 0 | 0 |

## *Definition (Interim Summary)*

Matrix $T$ with columns $A$, $B$ and $C$ encodes correct execution of the program, if the following two conditions hold:

1. $\forall i : A_i Q_{Li} + B_i Q_{Ri} + A_i B_i Q_{Mi} + C_i Q_{Oi} + Q_{ci} + \Pi_i = 0$
2. $\forall i, j, k, l : (V_{i,j} = V_{k,l}) \Rightarrow (T_{i,j} = T_{k,l})$

# Polynomial Form

# Matrices to Polynomials

Encode matrices into a few equations on polynomials.

Let $\omega$ be a primitive $N$-th root of unity and let $\Omega = \{\omega^j : 0 \leq j < N\}$. Let $a, b, c, q_L, q_R, q_M, q_O, q_C, \pi$ be polynomials of degree at most $N$ that interpolate corresponding columns from matrices at the domain $\Omega$. This means, that $\forall j : a(\omega^j) = A_j$ and the same for other columns.

***Proposition***

Now we can reduce down our first condition to checking valid execution trace into the following claim over polynomials:

$$\exists t \in \mathbb{F}[X] : \; aq_L + bq_R + abq_M + cq_O + q_C + \pi = z_\Omega t,$$

where $z_\Omega(X)$ is the vanishing polynomial $X^N - 1$.

# Copy constraints in polynomial form.

Spoiler: we can use the concept of permutation to encode $V$ wirings.

A permutation is a rearrangement of the set:

$$\mathcal{I} = \{(i, j) : \text{such that } 0 \leq i < N, \text{ and } 0 \leq j < 3\}$$

Permutation of the set is commonly denoted as $\sigma$.

# Copy constraints in polynomial form.

*Example*

The matrix $V$ induces a permutation of this set where $\sigma((i,j))$ is equal to the indices of the next occurrence of the value at position $(i,j)$. So, for our example:

$V$:

| L | R | O |
|---|---|---|
| 0 | | |
| 1 | | |
| 2 | 0 | 3 |
| 1 | 3 | |

$$\sigma((0,0)) = (2,1), \sigma((0,1)) = (0,3), \sigma((0,2)) = (0,2)$$

$$\sigma((0,3)) = (0,1), \sigma((2,1)) = (0,0), \sigma((3,1)) = (2,2)$$

**Permutation Check.** Having defined permutation, we can now reduce a condition 2 of valid execution trace matrix into the following check:

$$\forall (i,j) \in \mathcal{I} : T_{i,j} = T_{\sigma(i,j)}$$

You may have noticed how this can be reformulated as equality of $A$ and $B$:

$$A = \{((i,j), T_{i,j}) : (i,j) \in \mathcal{I}\}$$

$$B = \{(\sigma((i,j)), T_{i,j}) : (i,j) \in \mathcal{I}\}$$

We can reduce this check down to polynomial equations.

Suppose we have sets $A = \{a_0, a_1\}$ and $B = \{b_0, b_1\}$. We can consider polynomials $A' = \{a_0 + X, a_1 + X\}, B' = \{b_0 + X, b_1 + X\}$. So, $A' = B'$, only if $(a_0 + X)(a_1 + X) = (b_0 + X)(b_1 + X)$. This is true because of linear polynomial unique factorization property, working as prime factors. Now, we can utilize Schwartz-Zippel lemma to replace the latter formula with $(a_0 + \gamma)(a_1 + \gamma) = (b_0 + \gamma)(b_1 + \gamma)$ for some random $\gamma$ with overwhelming probability. If we wish to generalize this for arbitrary sets $A = \{a_0, \dots, a_{k-1}\}$ and $B = \{b_0, \dots, b_{k-1}\}$, apply the following check:

$$\prod_{i=0}^{k-1}(a_i + \gamma) = \prod_{i=0}^{k-1}(b_i + \gamma)$$

Let $\Omega$ be a domain of the form $\{1, \omega, \ldots, \omega^{k-1}\}$ for some $k$-th root of unity $\omega$. Let $f$ and $g$ be polynomials that interpolate the following values at $\Omega$:

$$f : (a_0 + \gamma, \ldots, a_{k-1} + \gamma)$$

$$g : (b_0 + \gamma, \ldots, b_{k-1} + \gamma)$$

Then $\prod_{i=0}^{k-1}(a_i + \gamma) = \prod_{i=0}^{k-1}(b_i + \gamma)$ if and only if exists a polynomial $Z \in \mathbb{F}[X]$ such that for all $h \in \Omega$ we have $Z(\omega^0) = 1$ and $Z(h)f(h) = g(h)Z(\omega h)$.

Now that we can encode equality of sets of field elements, let's expand this to sets of tuples of field elements. Let $A = \{(a_0, a_1), (a_2, a_3)\}$ and $B = \{(b_0, b_1), (b_2, b_3)\}$, then, similarly:

$$A' = \{a_0 + a_1 Y + X, a_2 + a_3 Y + X\}$$

$$B' = \{b_0 + b_1 Y + X, b_2 + b_3 Y + X\}$$

$$A = B \leftrightarrow A' = B'$$

As before, we can leverage Schwartz-Zippel lemma to reduce this down into sampling random $\beta$ and $\gamma$ and checking equality of:

$$(a_0 + \beta a_1 + \gamma)(a_2 + \beta a_3 + \gamma) = (b_0 + \beta b_1 + \gamma)(b_2 + \beta b_3 + \gamma)$$

Let's make $(i, j)$ into one field element, so that we can use statement above for encoding.

Recall that $i \in [0; N-1]$ and $j \in [0; 2]$; we can take $3N$-th primitive root of unity $\eta$ and define our field element as $a_0 = \eta^{3i+j}$:

$$A = \{(\eta^{3i+j}, T_{i,j}) : (i, j) \in \mathcal{I}\}$$

$$B = \{(\eta^{3k+l}, T_{i,j}) : (i, j) \in \mathcal{I}, \sigma((i, j)) = (k, l)\}$$

Let $\eta$ be a $3N$-th primitive root of unity, $\beta$ and $\gamma$ random field elements. Let $\mathcal{D} = \{1, \eta, \eta^2, \ldots, \eta^{3N-1}\}$. Then let $f$ and $g$ interpolate at $\mathcal{D}$:

$$f : \{T_{i,j} + \eta^{3i+j}\beta + \gamma : (i,j) \in \mathcal{I}\}$$
$$g : \{T_{i,j} + \eta^{3k+l}\beta + \gamma : (i,j) \in \mathcal{I}, \sigma((i,j)) = (k,l)\}$$

So, $\exists Z \in \mathbb{F}[X]$ s.t. $\forall h \in \Omega$ we have $Z(\eta^0) = 1$ and $Z(h)f(h) = g(h)Z(\eta h) \leftrightarrow A = B$ w.h.p.

Notice, that $\omega = \eta^3$ is a primitive $N$-th root of unity. Let $\Omega = \{1, \omega, \omega^2, \ldots, \omega^{N-1}\}$. We will define three polynomials, which interpolate following sets:

$$S_{\sigma 1} : \{\eta^{3k+l} : (i, 0) \in \mathcal{I}, \sigma((i, 0)) = (k, l)\}$$

$$S_{\sigma 2} : \{\eta^{3k+l} : (i, 1) \in \mathcal{I}, \sigma((i, 0)) = (k, l)\}$$

$$S_{\sigma 3} : \{\eta^{3k+l} : (i, 2) \in \mathcal{I}, \sigma((i, 0)) = (k, l)\}$$

# Copy constraints via polynomials

Let $\omega$ be an $N$-th root of unity. Let $\Omega = \{1, \omega, \omega^2, \ldots, \omega^{N-1}\}$. Let $k_1$ and $k_2$ be two field elements such that $\omega^i \neq \omega^j k_1 \neq \omega^l k_2$ for all $i, j, l$. Let $\beta$ and $\gamma$ be random field elements. Let $f$ and $g$ be the polynomials that interpolate, respectively, the following values at $\Omega$:

$$f : \left\{ \left(T_{0,j} + \omega^i \beta + \gamma\right) \left(T_{1,j} + \omega^i k_1 \beta + \gamma\right) \left(T_{2,j} + \omega^i k_2 \beta + \gamma\right) : 0 \leq i < N$$

$$g : \left\{ \left(T_{0,j} + S_{0,1}(\omega^i)\beta + \gamma\right) \left(T_{0,j} + S_{0,2}(\omega^i)\beta + \gamma\right) \left(T_{0,j} + S_{0,3}(\omega^i)\beta + \gamma\right)$$

So, $\exists Z \in \mathbb{F}[X]$ such that $\forall d \in \mathcal{D}$ we have $Z(\omega^0) = 1$ and $Z(d)f(d) = g(d)Z(\omega d) \leftrightarrow A = B$ w.h.p.

# Summary | Matrices

**Definition**

Let $T$ be a $N \times 3$ matrix with columns $A, B, C$ and $\Pi$ a $N \times 1$ matrix where $N$ is the number of gates. They correspond to a valid execution instance with public input given by $\Pi$ if and only if:

1. $\forall i : A_i Q_{Li} + B_i Q_{Ri} + A_i B_i Q_{Mi} + C_i Q_{Oi} + Q_{Ci} + \Pi_i = 0$
2. $\forall i, j, k, l : V_{i,j} = V_{k,l} \implies T_{i,j} = T_{k,l}$
3. $\forall i > n : \Pi_i = 0$

# Summary | Polynomials

***Definition***

Let $z_\Omega = X^N - 1$. Let $T$ be a $N \times 3$ matrix with columns $A, B, C$ and $\Pi$ a $N \times 1$ matrix. They correspond to a valid execution instance with public input given by $\Pi$ if and only if:

1. $\exists t_1 \in \mathbb{F}[X] : aq_L + bq_R + abq_M + cq_O + q_C + \pi = z_\Omega t_1$
2. $\exists t_2, t_3, z \in \mathbb{F}[X] : zf - gz' = z_\Omega t_2$ and $(z-1)L_1 = z_\Omega t_3$, where $z'(X) = z(X\omega)$.

# Thank you for your attention

♥

zkdl-camp.github.io
github.com/ZKDL-Camp