Hello!

We are adding Halo2 functionality to Noir. Originally, we explored the laborious task of synthesizing the Halo2 matrix representation from each ACIR opcode (like VampIR[1]). However, “UltraPlonK” is “PLONKish” arithmetisation, which prompts the idea that we could simply use Barretenberg to build the polynomials that encode the circuit. Then, we can use the Halo2 API to construct the vanishing argument, multipoint opening argument, and inner product argument prescribed to make a Halo2 proof.

This can be quickly visualized using the high-level description of the Halo2 Proving System:

[

Halo2 + Barretenberg

774×529 167 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/2e7eaaafeb34d3d8ed4d1ce64b81412a73bb5e8f.jpeg)

The biggest (only?) impediment to native interoperability in arithmetisation is how Halo2 handles the blinding/ “Zero Knowledge adjustment” of:

- Lookup Arguments
- Permutation Arguments [2]

Assuming proper ZK adjustments, this could be a strong tailwind for the integration of Halo2 into Noir since we would rely on existing “Black Box Function” [3] OpCodes rather than connecting existing Halo2 gadgets and building others. This further begs the question as to whether we even need to build a “backend” for Noir. Instead, it appears that we could potentially work entirely within the Barretenberg library by adding IPA commitments.

Questions:

- How does Barretenberg handle blinding of Lookup and Permutation arguments?
- Do we need to make serious changes to fulfill compatibility with Halo2’s Zero Knowledge adjustment?
- Do we need to make serious changes to fulfill compatibility with Halo2’s Zero Knowledge adjustment?
- Instead of building a new backend for Noir, should we extend the Barretenberg library to use IPA if desired?
- Should we still bind to the Halo2 sdk for the remaining Halo2 proof system steps so that we do not write a lot of code that would need analysis/ auditing? Does this require a backend, or can we comfortably add FFI C++ → Rust bindings to the Barretenberg library?
- Should we still bind to the Halo2 sdk for the remaining Halo2 proof system steps so that we do not write a lot of code that would need analysis/ auditing? Does this require a backend, or can we comfortably add FFI C++ → Rust bindings to the Barretenberg library?
- Are there any other concerns/ issues we are overlooking?

Bonus questions on KZG:

- It appears that Barretenberg already makes KZG commitments[4] in proofs to some degree. How different is this from the PSE fork of Halo2 which swaps IPA for KZG?

Additional Links

replace “(dot)” with “.” (only 2 allowed in post for new user)