Last week we made a v0.2 release ofMiden VM (the VM around which we are building Polygon Miden zk-rollup). The v0.1 release was largely based on Distaff VM architecture, but v0.2 is a complete overhaul which comes with lots of new features and makes the VM Turing-complete.

New architecture

A lot of things have changed under the hood in this release. To mention a few:

- We migrated proof generation from a 128-bit field to a 64-bit field. Specifically, we now use a prime field with characteristic 2^{64} - 2^{32} + 1

(the same field is used by Polygon Zero, Polygon Hermez, and a couple other projects). This field has several really cool properties (enabling very fast proof generation being one of them).

- We adopted chiplet-based architecture, where some complex computations are “offloaded” to specialized chiplets

(i.e., each chiplet is described by its own set of AIR constraints). These chiplets are then tied to the main VM via multiset checks, similar to the ones used inPLONK (for Plookup) and in Cairo (for builtins).

- We redesigned how we construct Merkelized Abstract Syntax Trees of programs. Now,program MAST is a true binary tree where internal nodes define control flow, and leaves contain sequential instructions with no control flow. Hashes of leaf nodes can be computed very efficiently - i.e., we can hash up to 72 instructions in a single permutation of Rescue Prime hash function.
- We redesigned program decoder (which is responsible for traversing program MAST) to remove all limitations on the depth of control flow logic.
- We removed restrictions on the depth of theoperand stack by introducing an overflow table (implemented via multiset checks). This table keeps track of all elements of the stack which didn’t fit into the top 16 slots.

The layout of the VM’s execution trace looks as follows:

[

1694×933 107 KB

](https://ethresear.ch/uploads/default/original/2X/3/33b109f5aca3b62e46f8366f887ba5e8aa41053d.png)

In the above, virtual tables

and buses

are variations on multiset checks mentioned previously. The detailed explanation of the design (including all AIR constraints) can be found here.

Overall, execution trace of the VM requires 84 columns for ~100-bit security level and 93 columns for ~128-bit security level (each column contains 64-bit values). We anticipate adding 5 - 10 more columns as we add more features to the VM (planned for v0.3 release) - so, the final column count will be closer to 100. Maximum degree of AIR constraints we use in the VM is 9.

New features

The new architecture allowed us to make the VM much more expressive. Specifically:

- As already mentioned above, the new program structure and decoder design allowed us to remove all restrictions on the depth of the control flow logic.
- We added random access memory to the VM. Specifically, Miden VM support traditional read-write RAM which is actually one of the chiplets as described here.
- We added support for operations over regular 32-bit unsigned integers. The new field allowed us to implement such operations pretty efficiently (see here for the detailed description). Thus, Miden VM could be viewed as a regular 32-bit VM.
- Because of the chiplet-based design, we can add pretty powerful instructions to the VM. One example of this is an instruction which can verify a Merkle path in a single VM cycle.

We’ve also made quite a few improvements to theMiden assembly language, which now support procedures with local variables and comes with a standard library (which is, for now, fairly nascent).

Performance

A really nice benefit of using a small field and adopting a chiplet-based design is that, even after adding all of the above features, performance of the VM actually improved. Here are a few benchmarks for executing the VM for 2^{20}

cycles (using BLAKE3 as the hash function and targeting 96-bit security level):

Machine

Execution time

Proving time

Apple M1 Pro (1 thread)

330 ms

79 sec

Apple M1 Pro (8 threads)

330 ms

12.8 sec

Amazon Graviton 3 (64 threads)

390 ms

3.2 sec

Proof size for this computation is about 100 KB, and verification time is really fast - usually under 2 ms.

To give some ideas of what can be done in 2^{20}

cycles:

- A 2-to-1 BLAKE3 hash can be computed using about 5K VM cycles (the implementation ishere).
- A 2-to-1 SHA256 has can be computed using about 15K VM cycles (the implementation ishere).

Thus, for example, on an M1 Pro Mac, the VM can execute (and prove) over 5 SHA256 hashes per second. This is without any specialized circuits or instructions.

Also, there is still quite a bit of room left for optimizing both the prover and the implementation of the primitives - so, we hope that the above numbers will improve in the future.

Next steps

Miden VM is still a work-in-progress. As mentioned above, we have a few important features planned for the next release. These include:

- Function calls with logically separated memory spaces.
- Customizable VM kernels.
- Efficient way for supporting a large number of public inputs/outputs.

But we'd also like to know what others think - so, we welcome any feedback about the design or the implementation (the VM is fully open-source - so, feel free to take a look at the code).