I've started working on libraries of STARK gadgets (basically, just AIR for different constructs written in AirAssembly). So far, I've put together just a few gadgets which are located here. There are currently 2 libraries: one for a 128-bit field and another one for a 224-bit field.

The current list of gadgets includes:

1. Poseidon hash function
2. this is a simple gadget which allows creating proofs for pre-image of Poseidon hash function.
3. Merkle path authentication
4. also a relatively simple gadget allows proving that you know a path to specific node in a Merkle tree. Uses Poseidon hash function.
5. Merkle tree update
6. a gadget that allows proving that a sparse Merkle tree was updated correctly (more info below).
7. Schnorr signature verification
8. this gadget is available only for the 224-bit field library, and I've described it in some detailshere.

Merkle tree update:

This is a new gadget I've put together just recently. It works as follows:

1. In the 224-bit version, the execution trace consists of 12 registers (or 24 registers in 128-bit version):

a. The first 6 registers are used to validate Merkle path to a given leaf using the pre-update value of the leaf.

b. The other 6 registers are used to validate Merkle path to the same leaf, but using the post-update value of the leaf.

1. If the update was done correctly, at the end of the execution trace, one of the registers will contain the old tree root, while another register will contain a new tree root. And these can be verified using boundary constraints.

You check out a working examplehere.

On my machine, it takes about 4 seconds to generate a proof of update in a tree of depth 16, and the proof size is about 105 KB. This is for a somewhat optimized version in a 128-bit field, the 224-bit version is much slower than that (primarily because for a 224-bit field all math happens in JavaScript).

In the current implementation, the STARK proves only that a Merkle tree was updated correctly. It is possible, with relatively little effort, to create a similar gadget but where the verifier would also be able to check the exact values of the leaf (before and after update).

Overall, there is still a lot that can be improved about the gadgets, both, in terms of functionality and efficiency. So, by all means, play around withe them - but don't use them in production.

If anyone has thoughts about other gadgets that might be useful (or about improvements to the current ones) - feedback is always welcome!