Review of Empty Sector Update Circuit

Protocol Labs

J.P. Aumasson and Antony Vennard

20211210

Contents

1 Issues

1.1 `trailing_zero()` values

Many of the `trailing_zeros()` alls seem to assume that the value is a power of 2. This is fine if all values are powers of 2, but will be risky if not.

Examples would be:

- constants.rs line 66.

Answer: The Merkle trees are all chosen via type parameters, but the code assumes that `TreeD` Is always a arity-2 tree.

1.2 Defensive coding: additional bounds checks

- In this location gadgets.rs line 188

  h is assumed to be in [1,bit_len-1]. Would it be worthwhile to do bounds checking?

  The values of h are hardcoded so the risk is low.

- Similarly, bounds checks might be worthwhile here: gadgets.rs line 96-97.

Answer: We will implement this.

1.3 Possibly lossy cast:

Here, can the cast be lossy: gadgets.rs line 103?

Answer: Only if the gadget caller passed in values for challenges and bits_per_challenge which weren't computed using constants.rs.

We will consider if an additional check is possible.

1.4 True Randomness versus PRF definition.

Unless I miss sth, the "random" bits from `gen_challenge_bits()` are not actually random, but derived (deterministically) and unpredictable unless you know all the PRF inputs

1.5 Use of asserts: some may crash.

For example in `PublicInputs::new`, the `assert!` macro will cause a `panic` in both debug and release public builds, which will crash any process or thread it is part of.

It may be worth converting such statements either to `std::Result` returns, or if an assert is desired, `debug_assert!`, which will only panic in debug modes.

Answer: We will go through all of the asserts to check them. Some would probably be better implemented as results in another way.

However some asserts, e.g. checks of system parameters, would mean that a circuit could not be constructed at all and indicate that the software has been incorrectly compiled (e.g. by changing the constants). So we will keep these as `assert!`s.

2 Questions and Comments

2.1 SectorNodes clarification

Is `sector_nodes` or `SectorNodes` always 2^30 or 2^31? constants.rs line 66

Answer: In production yes. For test and for development, we allow smaller values.

2.2 Move cheaper asserts before circuit evaluations if possible.

It may be interesting to move these cheap asserts before the more complex k and h validations https://github.com/filecoin-project/rust-fil-proofs/blob/f71c113f14c03ec6a966199cd32539edcb4db153/storage-proofs-update/src/circuit.rs#L403-L406

Answer: We like this suggestion.

2.3 Poseidon input from pairings

will the Fr-s hashed by Poseidon be of constant size? circuit.rs lines 491-494

Answer: Fr is a fixed size and changing it would be a major endeavour.

2.4 Rust dereference issue

Shouldnt this be `*bit` instead of '`**bit`? to count the non-zeros circuit.rs line 393

Answer (by ourselves): This is required as the lambda borrows from a borrow, as in the following example:

```
fn main() {

    let boolvec : Vec<bool> = vec![true, false, true, false, false];

    let borrowed_boolvec = &boolvec;
    let only_true = borrowed_boolvec.iter().filter(|v| v);
    for v in only_true {
        println!("Result is {}", v)
    }
}
```