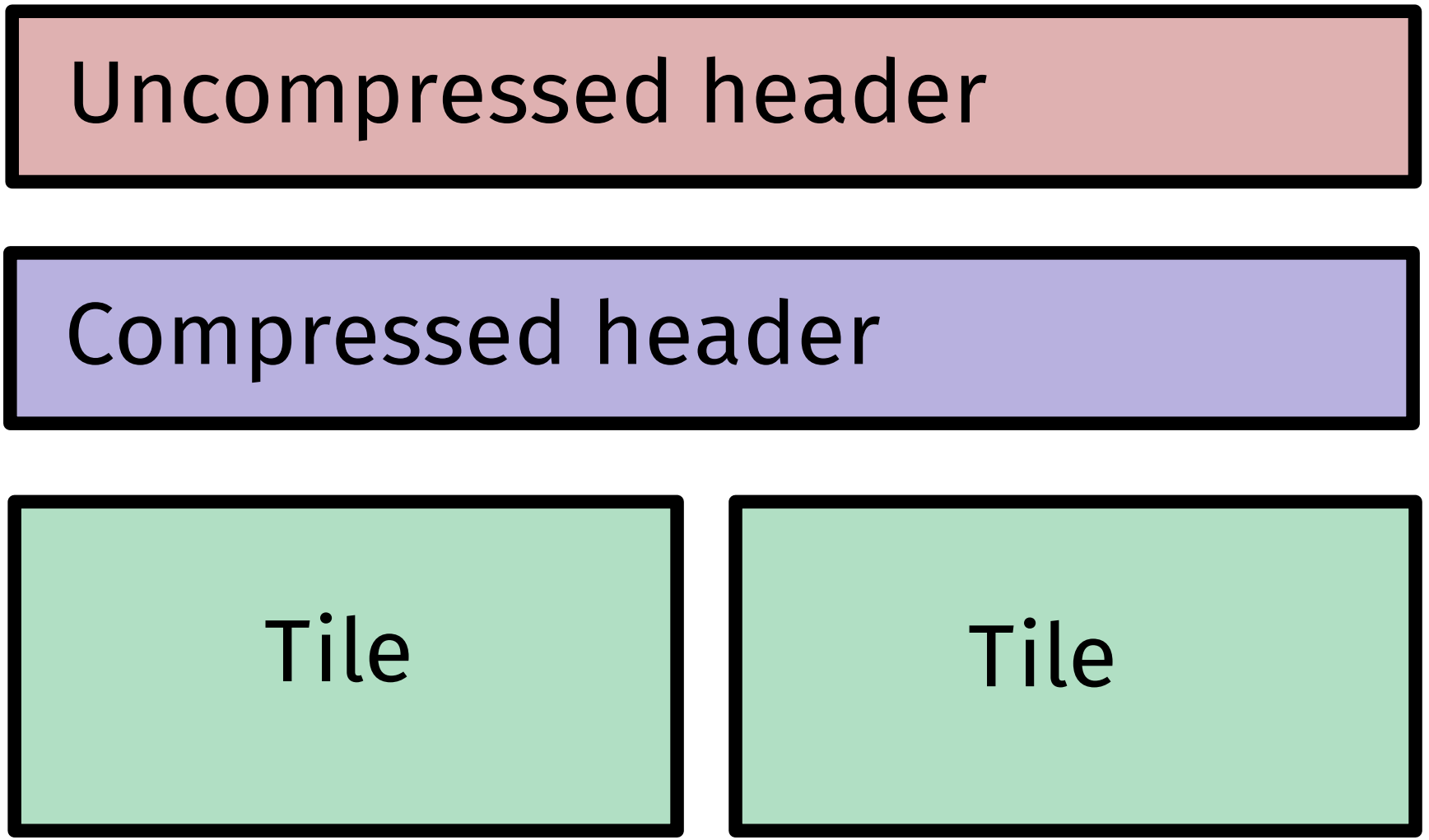
The fastest and worstest AV1 encoder

Thomas Daede

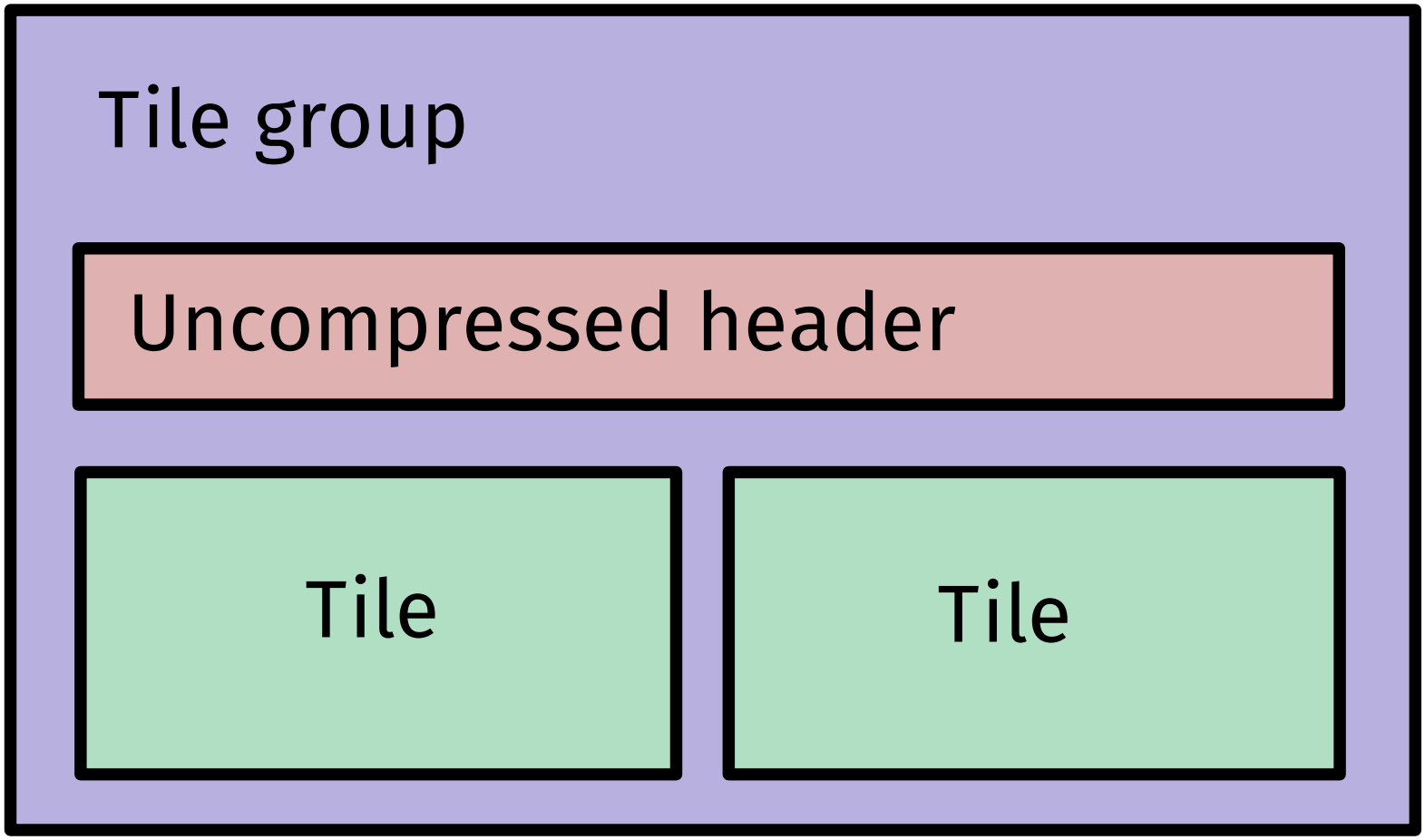
but why

- Started as a reimplementation of AV1 in order to find bitstream and specification bugs
- Could do an encoder or decoder:
  - Decoder (especially fuzzed) more useful to find mismatches
  - Encoder doesn’t need all features implemented to work
- Algorithmic improvements over libaom

VP9 frame at a glance



AV1 frame at a glance



The compressed header is gone!

- In VP9, probabilities were stored in the compressed header and remained constant for the frame.
  - Encode without writing the bitstream, but count the probabilities for each symbol
  - Write probabilities in compressed header
  - Write actual bitstream using probabilities in the compressed header

Tile

DC	DC	DC	H_PRED
1	1	1	01

Compressed header

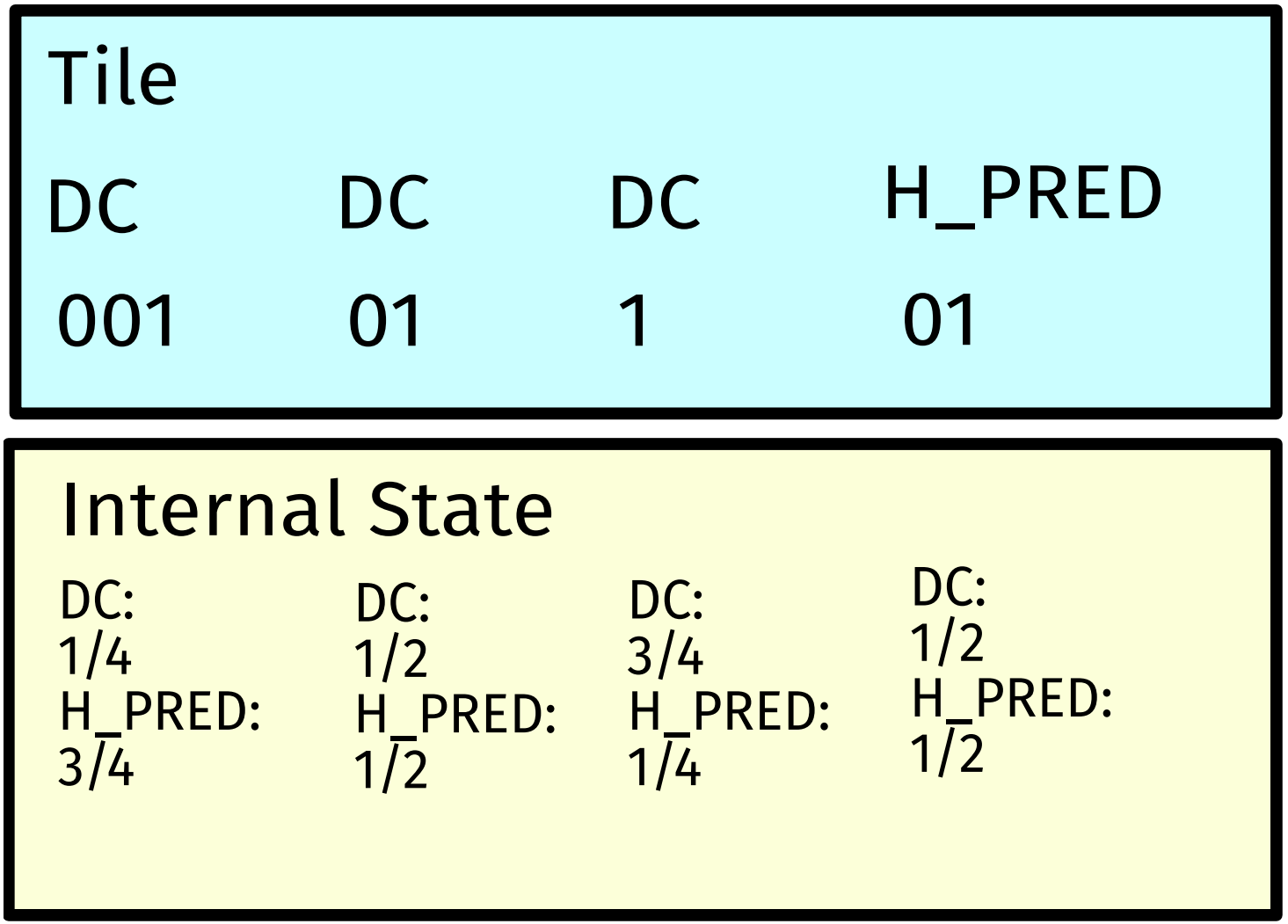
DC = 3/4
H_PRED = 1/4

Downsides

- Probabilities during RDO aren’t the real probabilities because you haven’t counted yet
- Have to buffer coefficients or tokens in memory

AV1

- Probabilities updated every symbol
- Exact rate known simply by coding the symbol
- RDO can be done by coding the symbol, then rewinding
- Once a superblock is encoded, it never has to be touched again (except for loop filter)


Tile
DC
DC
DC
H_PRED
001
01
1
01
Internal State
DC:
1/4
H_PRED:
3/4
DC:
1/2
H_PRED:
1/2
DC:
3/4
H_PRED:
1/4
DC:
1/2
H_PRED:
1/2

Downsides

- Internal state needs to be saved and restored when doing RDO (make a decision, undo it)
- State is intentionally kept small, but still 100kB+
- Need clever way of only saving and restoring the required parts

It’s in Rust

- Read Kostya’s blog post for details

Rust: Not So Great for Codec Implementing

It’s in Rust

- Read Kostya’s blog post for details

Rust: Not So Great for Codec Implementing

Overall, Rust is not that bad and I’ll keep developing NihAV using it but keep in mind it’s still far from being perfect (maybe about as far as C but in a different direction).

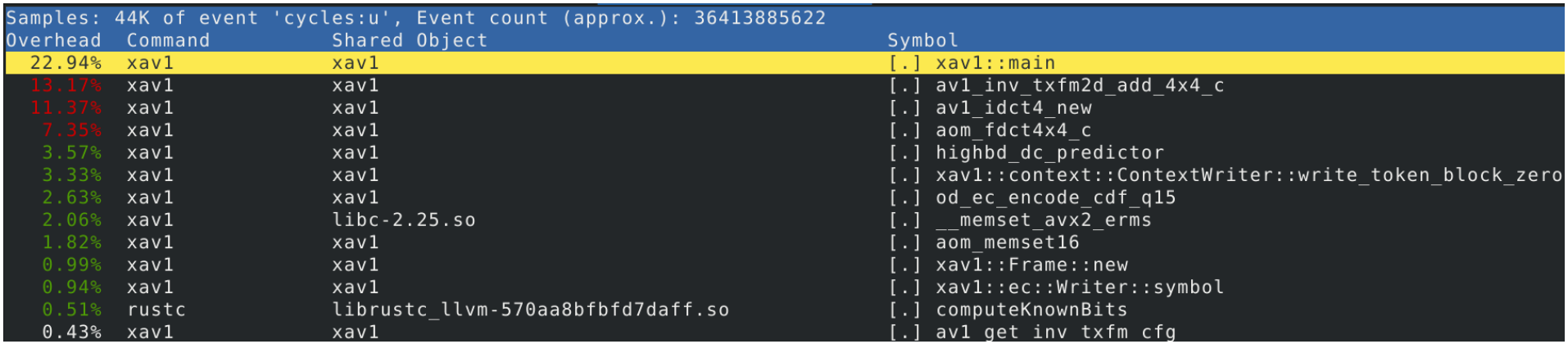
It’s in Rust

- Read Kostya’s blog post for details

Rust: Not So Great for Codec Implementing

Overall, Rust is not that bad and I’ll keep developing NihAV using it but keep in mind it’s still far from being perfect (maybe about as far as C but in a different direction).

Overall, Rust is not that bad


Samples: 44K of event 'cycles:u', Event count (approx.): 36413885622
Overhead Command Shared Object Symbol
22.94% xav1 xav1 [.] xav1::main
13.17% xav1 xav1 [.] av1_inv_txfm2d_add_4x4_c
11.37% xav1 xav1 [.] av1_idct4_new
7.35% xav1 xav1 [.] aom_fdct4x4_c
3.57% xav1 xav1 [.] highbd_dc_predictor
3.33% xav1 xav1 [.] xav1::context::ContextWriter::write_token_block_zero
2.63% xav1 xav1 [.] od_ec_encode_cdf_q15
2.06% xav1 libc-2.25.so [.] __memset_avx2_erms
1.82% xav1 xav1 [.] aom_memset16
0.99% xav1 xav1 [.] xav1::Frame::new
0.94% xav1 xav1 [.] xav1::ec::Writer::symbol
0.51% rustc librustc_llvm-570aa8bfbfd7daff.so [.] computeKnownBits
0.43% xav1 xav1 [.] av1_get_inv_txfm_cfg

Some parts are still from libaom :(

- Transforms
  - Not final, not worth spending time on
- Predictions
- daala_ec
- Tables, etc

It’s in git

- https://github.com/tdaede/rav1e.git

Limitations

- Requires particular libaom git commit and configuration
- Intra only, 4x4 transforms
- Coefficient coder can’t code large values