libswscale reimagined

VDD'25

Niklas Haas

About Me

- **Niklas Haas (haasn)** <contact@niklashaas.de>
- Independent Consultant
- libplacebo, mpv, FFmpeg, dav1d

About this project:

- Software in the Public Interest, Inc.
- Sovereign Tech Fund 2024

libswscale, 2024

libswscale, 2024

- API is highly stateful:

```
av_opt_set_int(s, "srcw", inlink0->w, 0);
av_opt_set_int(s, "srch", inlink0->h, 0);
av_opt_set_int(s, "src_format", inlink0->format, 0);
av_opt_set_int(s, "dstw", outlink->w, 0);
av_opt_set_int(s, "dsth", outlink->h, 0);
av_opt_set_int(s, "dst_format", outfmt, 0);
// ...
av_opt_set_int(s, "src_h_chr_pos", scale->in_h_chr_pos, 0);
av_opt_set_int(s, "src_v_chr_pos", in_v_chr_pos, 0);
av_opt_set_int(s, "dst_h_chr_pos", scale->out_h_chr_pos, 0);
av_opt_set_int(s, "dst_v_chr_pos", out_v_chr_pos, 0);
```

libswscale, 2024

- API is highly stateful:

```
sws_getColorspaceDetails(s, (int **)&inv_table, &in_full,
                         (int **)&table, &out_full,
                         &brightness, &contrast, &saturation);

if (inlink->colorspace != AVCOL_SPC_UNSPECIFIED)
    inv_table = sws_getCoefficients(inlink->colorspace);
if (outlink->colorspace != AVCOL_SPC_UNSPECIFIED)
    table = sws_getCoefficients(outlink->colorspace);

sws_setColorspaceDetails(s, inv_table, in_full,
                         table, out_full,
                         brightness, contrast, saturation);
```

libswscale, 2024

- Cascaded internal mega-contexts:

```
typedef struct SwsContext {
    struct SwsContext *parent;

    AVSliceThread     *slicethread;
    struct SwsContext **slice_ctx;
    int               *slice_err;
    int              nb_slice_ctx;

    struct SwsContext *cascaded_context[3];
    int cascaded_tmpStride[4];
    uint8_t *cascaded_tmp[4];
    int cascaded_mainindex;

    // ...
} SwsContext;
```

libswscale, 2024

- Complex runtime functions:

```
static int scale_gamma(SwsContext *c, /*...*/)
{
    int ret = scale_internal(c->cascaded_context[0], /*...*/);
    if (ret < 0)
        return ret;

    if (c->cascaded_context[2])
        ret = scale_internal(c->cascaded_context[1], /*...*/);
    else
        ret = scale_internal(c->cascaded_context[1], /*...*/);
    if (ret < 0)
        return ret;

    if (c->cascaded_context[2])
        ret = scale_internal(c->cascaded_context[2], /*...*/);
    return ret;
}
```

libswscale, 2024

- Hundreds of bespoke conversion functions

```
needsDither = isAnyRGB(dstFormat) &&
        c->dstFormatBpp < 24 &&
        (c->dstFormatBpp < c->srcFormatBpp || (!isAnyRGB(srcFormat)));

/* yv12_to_nv12 */
if ((srcFormat == AV_PIX_FMT_YUV420P || srcFormat == AV_PIX_FMT_YUVA420P) &&
    (dstFormat == AV_PIX_FMT_NV12 || dstFormat == AV_PIX_FMT_NV21)) {
    c->convert_unscaled = planarToNv12Wrapper;
}
/* yv24_to_nv24 */
if ((srcFormat == AV_PIX_FMT_YUV444P || srcFormat == AV_PIX_FMT_YUVA444P) &&
    (dstFormat == AV_PIX_FMT_NV24 || dstFormat == AV_PIX_FMT_NV42)) {
    c->convert_unscaled = planarToNv24Wrapper;
}
/* nv12_to_yv12 */
if (dstFormat == AV_PIX_FMT_YUV420P &&
    (srcFormat == AV_PIX_FMT_NV12 || srcFormat == AV_PIX_FMT_NV21)) {
    c->convert_unscaled = nv12ToPlanarWrapper;
}
```

libswscale, 2024

- Highly divergent implementations

```
YUV420FUNC(yuv2rgb_c_48,     uint8_t,  0,  0, PUTRGB48, 48)
YUV420FUNC(yuv2rgb_c_bgr48,  uint8_t,  0,  0, PUTBGR48, 48)
YUV420FUNC(yuv2rgb_c_32,     uint32_t, 0,  0, PUTRGB,    8)
#if HAVE_BIGENDIAN
YUV420FUNC(yuva2argb_c,      uint32_t, 1, 24, PUTRGBA,   8)
YUV420FUNC(yuva2rgba_c,      uint32_t, 1,  0, PUTRGBA,   8)
#else
YUV420FUNC(yuva2rgba_c,      uint32_t, 1, 24, PUTRGBA,   8)
YUV420FUNC(yuva2argb_c,      uint32_t, 1,  0, PUTRGBA,   8)
#endif
YUV420FUNC(yuv2rgb_c_24_rgb, uint8_t,  0,  0, PUTRGB24, 24)
YUV420FUNC(yuv2rgb_c_24_bgr, uint8_t,  0,  0, PUTBGR24, 24)
YUV420FUNC_DITHER(yuv2rgb_c_16_ordered_dither, uint16_t, LOADDITHER16,  PUTRGB16,  8)
YUV420FUNC_DITHER(yuv2rgb_c_15_ordered_dither, uint16_t, LOADDITHER15,  PUTRGB15,  8)
YUV420FUNC_DITHER(yuv2rgb_c_12_ordered_dither, uint16_t, LOADDITHER12,  PUTRGB12,  8)
YUV420FUNC_DITHER(yuv2rgb_c_8_ordered_dither,  uint8_t,  LOADDITHER8,   PUTRGB8,   8)
YUV420FUNC_DITHER(yuv2rgb_c_4_ordered_dither,  uint8_t,  LOADDITHER4D,  PUTRGB4D,  4)
YUV420FUNC_DITHER(yuv2rgb_c_4b_ordered_dither, uint8_t,  LOADDITHER4DB, PUTRGB4DB, 8)
```

libswscale, 2024

- Unpredictable behavior
- API surprises
- Random bugs
- Difficult to extend

→ **Some cleanup required :)**

libswscale, 2025

libswscale, 2025

- Stateless public API:

```
typedef struct SwsContext {
    SwsFlags flags;
    SwsDither dither;
    SwsAlphaBlend alpha_blend;
    /* ... */
} SwsContext;

int sws_scale_frame(SwsContext *c, AVFrame *dst,
                    const AVFrame *src);
```

libswscale, 2025

- Stateless public API:

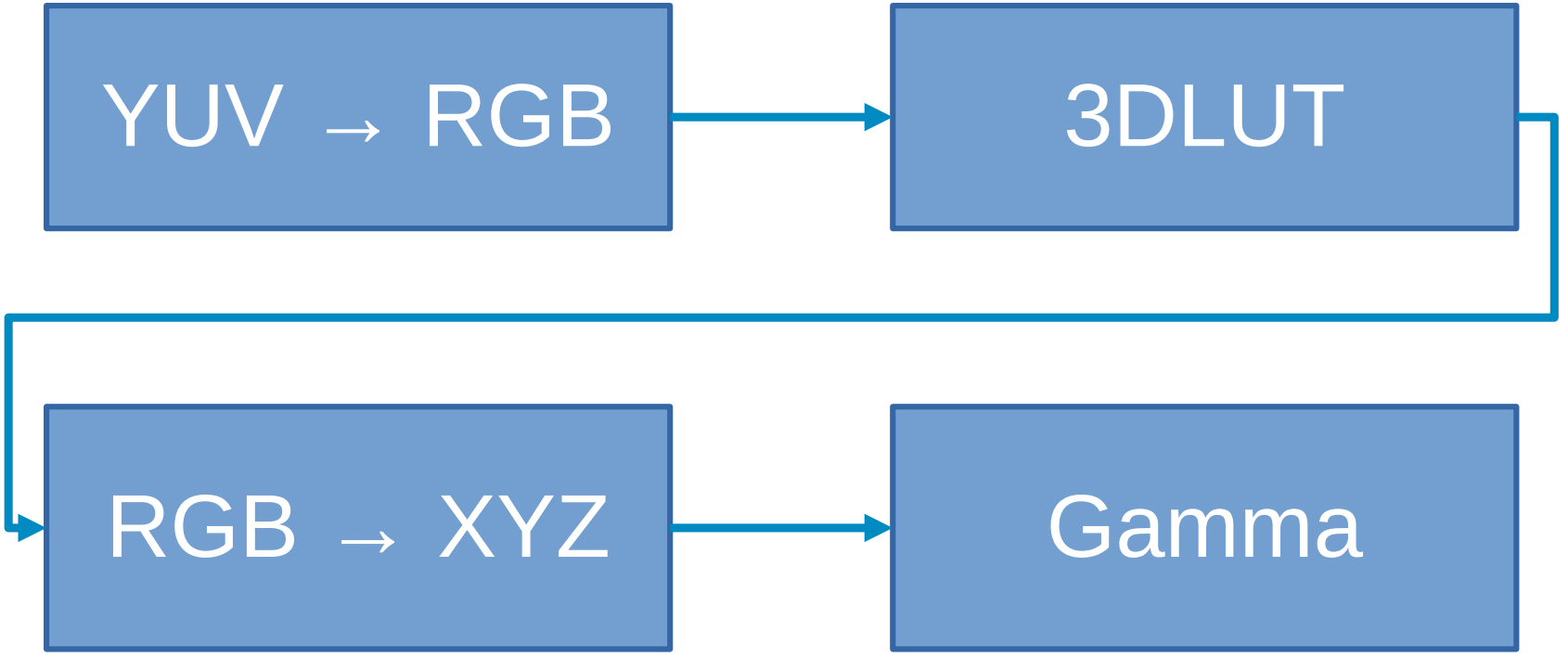
```
av_frame_copy_props(out, in);
out->width  = outlink->w;
out->height = outlink->h;
out->color_range = outlink->color_range;
out->colorspace = outlink->colorspace;
out->alpha_mode = outlink->alpha_mode;

int ret = sws_scale_frame(scale->sws, out, in);
```

libswscale, **2025**

- Frames
- Fields, formats
- Slices, passes
- Pixel conversions

libswscale, **2025**

- New helpers: **SwsGraph** + **SwsPass**



libswscale, **2025**

- **SwsPass** is self-contained
- Hides “legacy” sws contexts
- Graph recreated on demand
- Threading

- **Moves logic from run() to init()**

New format conversion

N·M problem

- 👎👎👎 N·M converters

- 👎 N input + M output
- 👎 No good common intermediate (YUV? RGB?)

- 👍 **N primitive operations**

SwsOp: Input/output

- **SWS_OP_READ**
- **SWS_OP_WRITE**
- **SWS_OP_SWAP_BYTES**
- **SWS_OP_(UN)PACK** (e.g. rgb565 : u16 → r5 g6 b5 : u8[3])

SwsOp: Pixel manipulation

- **SWS_OP_CLEAR** (e.g. rgbX → rgb0, y8 → yuv8)
- **SWS_OP_(L|R)SHIFT**
- **SWS_OP_SWIZZLE** (e.g argb → bgra)
- **SWS_OP_(CONVERT|EXPAND)** (e.g. u16 → float)
- **SWS_OP_DITHER**

SwsOp: Arithmetic operations

- **SWS_OP_LINEAR** (5x5 linear transform)
- **SWS_OP_SCALE** (scaler multiplication)
- **SWS_OP_(MIN|MAX)** (e.g. clamping)
- (defined on floats only)

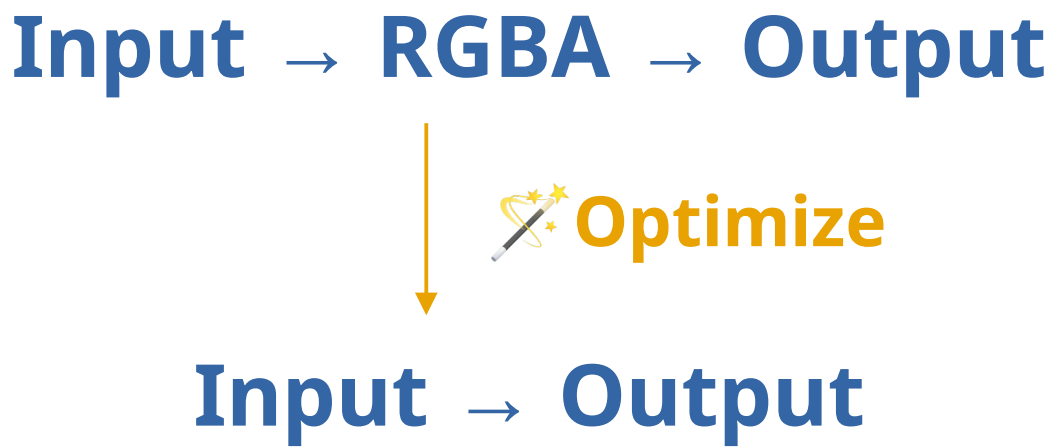
SwsOp: Usage

Single code path

SwsOp: Usage

Input → RGBA → Output

SwsOp: Usage



Example: y8 → rgb48

- **READ** : 1xU8 (planar)
- **CLEAR** : {_, 0, 0, 0}
- **CONVERT** : u8 → f32
- **LINEAR** : YUV8 → RGB16
- **MIN+MAX**: $(0 < x < 2^{16}-1)$
- **CONVERT**: float → u16
- **WRITE** : 3xU16 (packed)

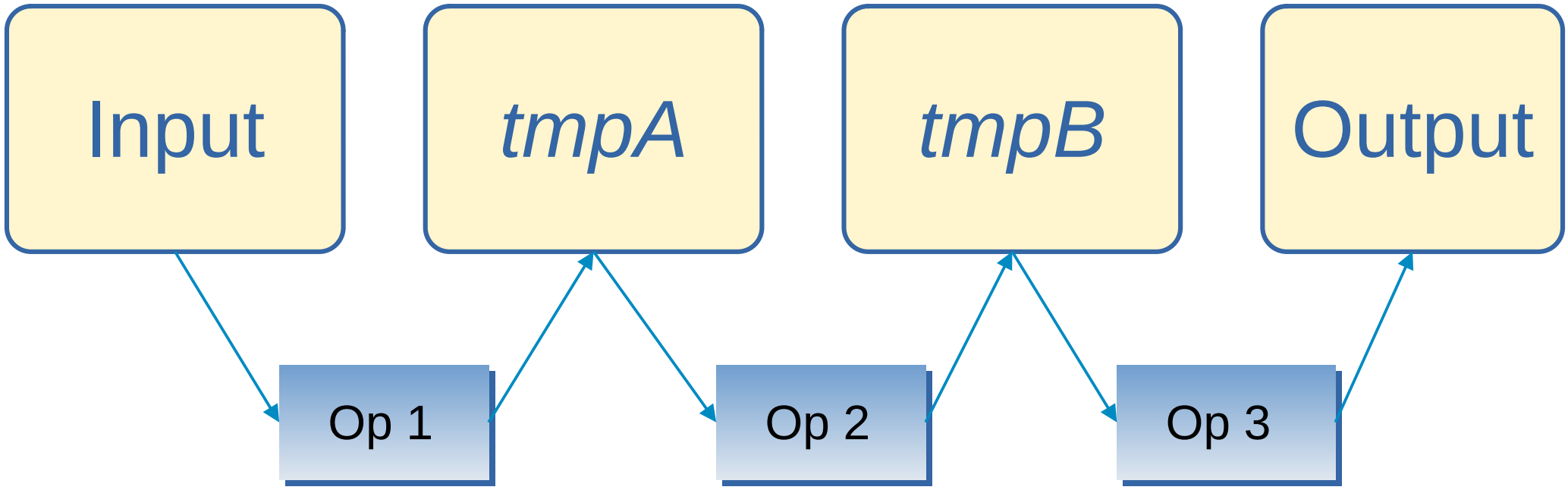
Example: y8 (gray) → rgb48

- **READ** : 1xU8 (planar)
- **CLEAR** : {_, 0, 0, 0}
- **CONVERT** : u8 → f32
- **LINEAR** : YUV8 → RGB16
- **MIN+MAX**: $(0 < x < 2^{16}-1)$
- **CONVERT**: float → u16
- **WRITE** : 3xU16 (packed)

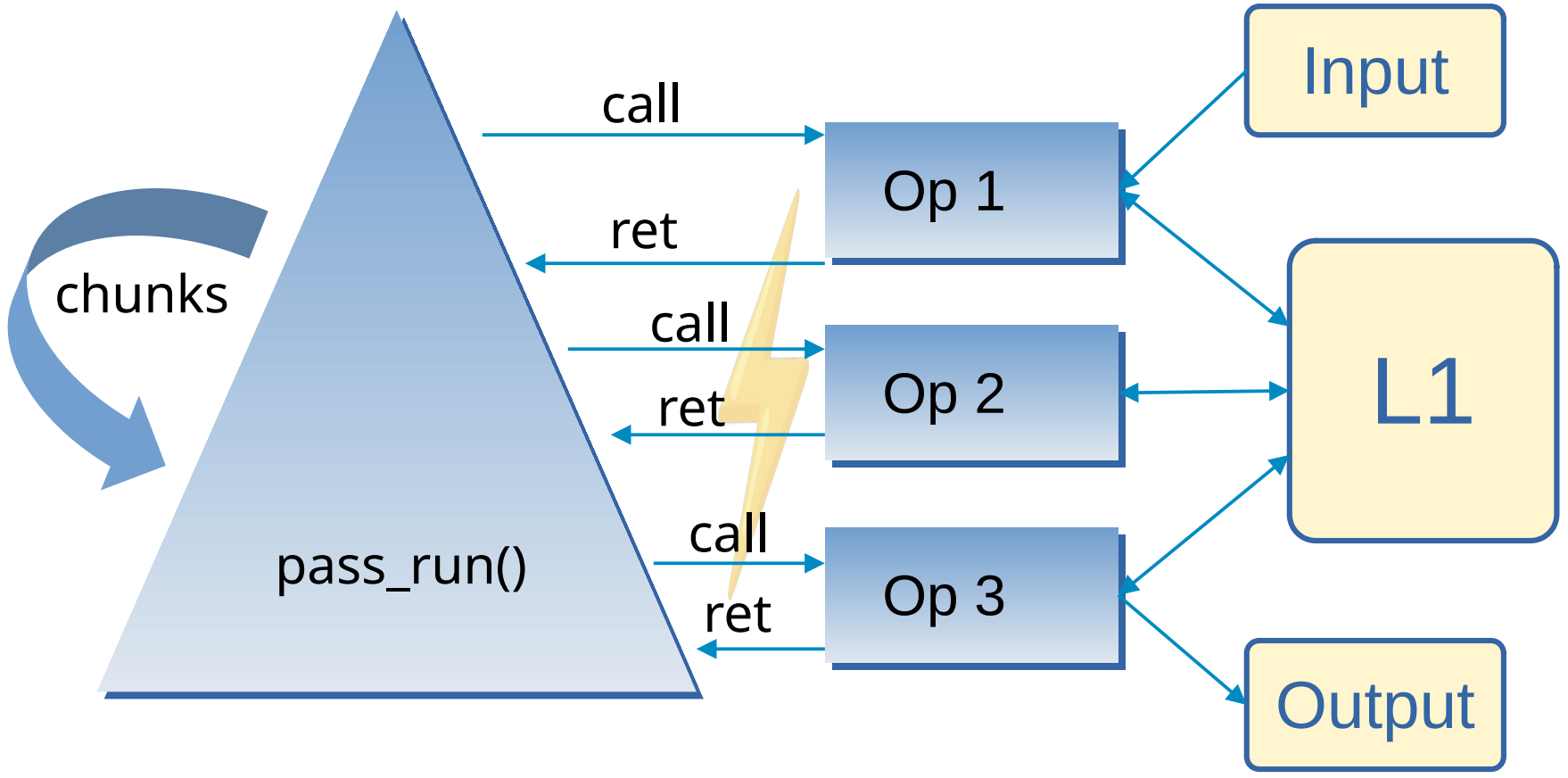
- **READ** : 1xU8 (planar)
- **EXPAND** : u8 → u16
- **SWIZZLE** : {0, 0, 0, 3}
- **WRITE** : 3xU16 (packed)

Implementation

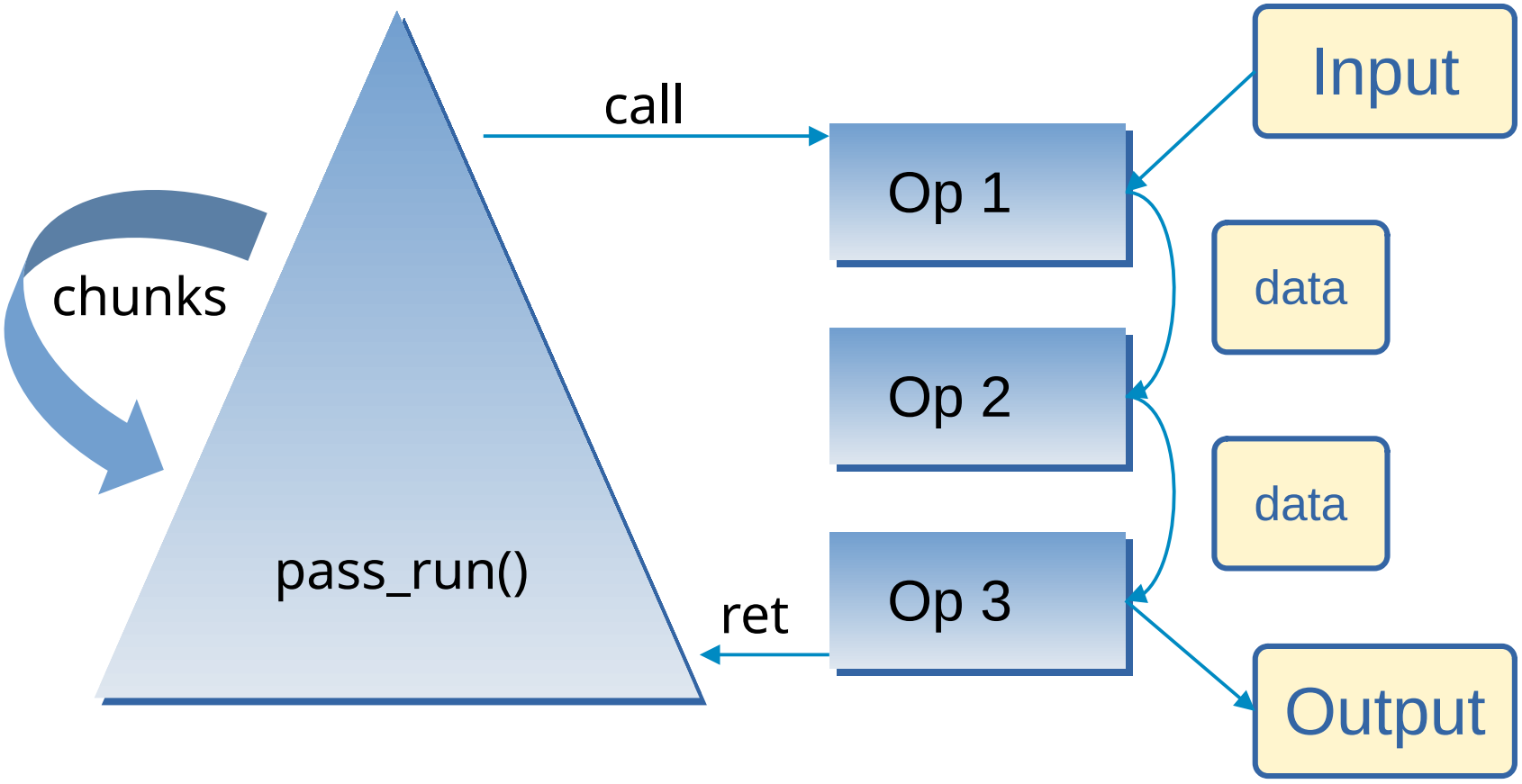
Naive



Iterate over chunks



Continuation passing style



Continuation passing style

```
void op_lshift(Op *next, pixel_t data[32])
{
    const int amount = op->data[0];

    for (int i = 0; i < 32; i++)
        data[i] <<= amount;

    next(&next[1], data);
}
```

x86 backend

- Pass data in vector registers directly

```
%macro lshift16 0
op lshift16
        vmovq xm8, [implq + SwsOpImpl.priv] ; load shift amount

IF X,   psllw mx, xm8 ; ← custom calling convention
IF Y,   psllw my, xm8
IF Z,   psllw mz, xm8
IF W,   psllw mw, xm8

        mov tmp0q, [implq + SwsOpImpl.cont]
        add implq, SwsOpImpl.next
        jmp tmp0q  ; jump to next operation
%endmacro
```

x86 backend

- 👍 Low overhead: 1 load, 1 add, 1 jmp
- 👍 Fully predicted
- 👍 No I/O
- 👍 SSE4, AVX, AVX2, AVX512, ...

- 👎 Redundant setup work
- 👎 Order swaps (pack/unpack)

Interlude

Packed shuffle solver

Packed shuffle solver

- **READ** : 1 planar element
- **CONVERT** : u8 → u16 (expand=True)
- **SWIZZLE** : {0, 0, 0, 3}
- **WRITE** : 3 packed elements

- Input: {YY} 8 bits
- Output: {YY YY YY YY YY YY} 48 bits

  → **pshufb** to the rescue

Packed shuffle solver

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

... *(apply all ops)*

→ {0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1}

→ Load as **pshufb** mask

Packed shuffle solver

```
.loop:
                movu m0, [srcq + srcidxq]
                pshufb m0, m1
                movu [dstq + dstidxq], m0
                add srcidxq, %1
IF %1 != %2,    add dstidxq, %2
                jnz .loop
                add srcq, src_strideq
                add dstq, dst_strideq
                ; ...
```

libswscale reimagined

Benchmarks

Benchmarks

Overall speedup=**3.390x** faster

(Ryzen 9 9950X3D, 16 threads)

Benchmarks

yuv444p → rgb24

speedup=**4.781x** faster

(127 µs vs 609 µs)

(Ryzen 9 9950X3D, 16 threads)

Honorable mention

bgr4_byte → rgb4_byte

speedup=**758.817x** faster

(20 µs vs 15221 µs)

(Ryzen 9 9950X3D, 16 threads)

Dishonorable mention

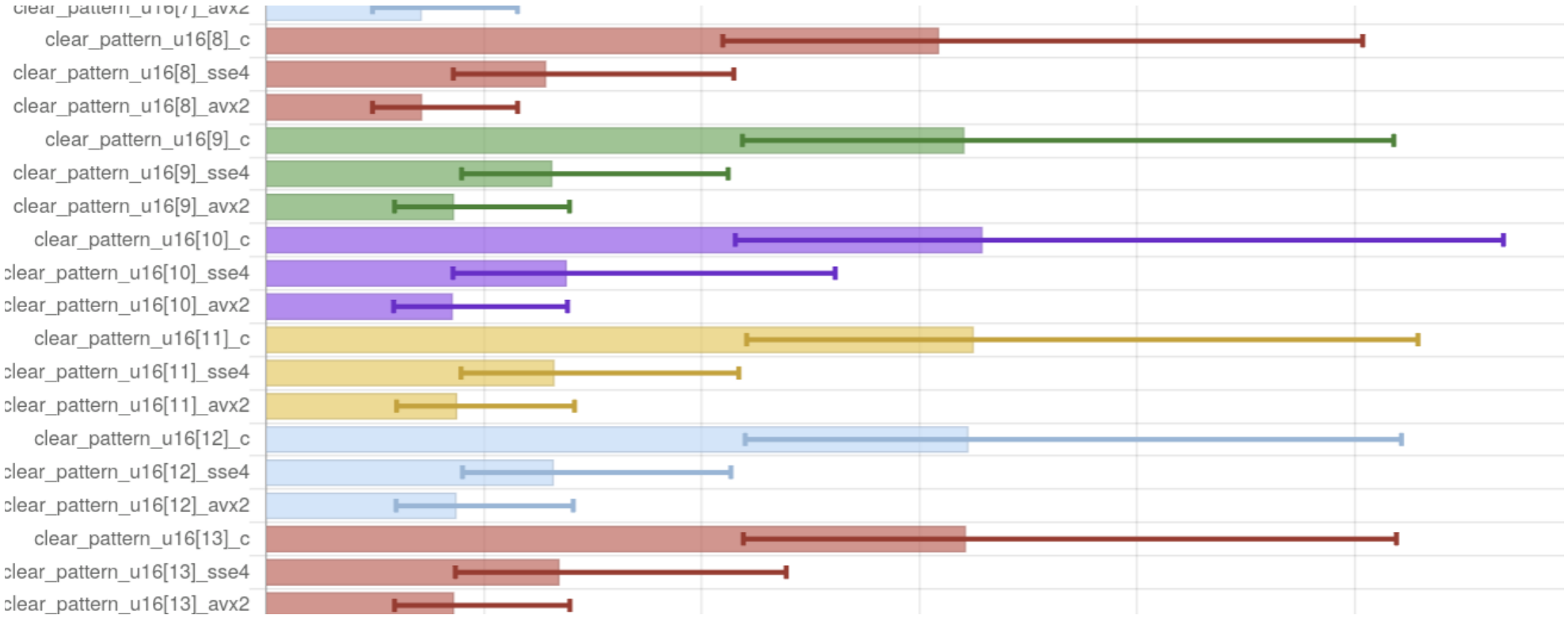
bgr555le → rgb565le

speedup=0.131x *slower*

(246 µs vs 41 µs)

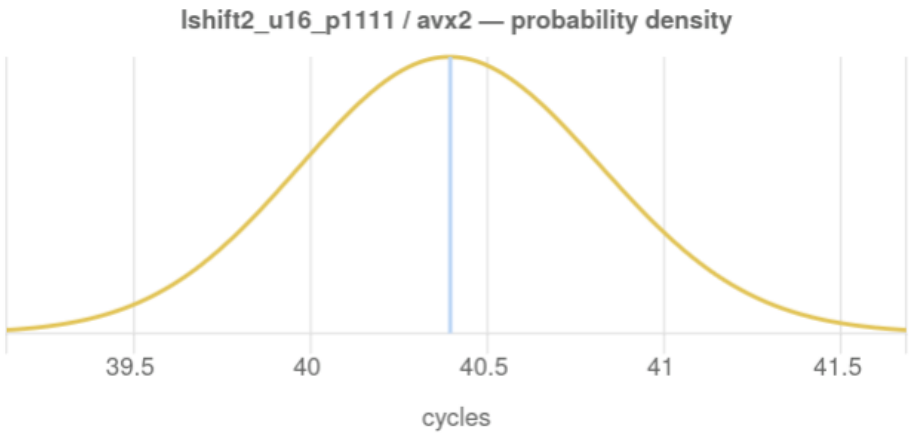
(Ryzen 9 9950X3D, 16 threads)

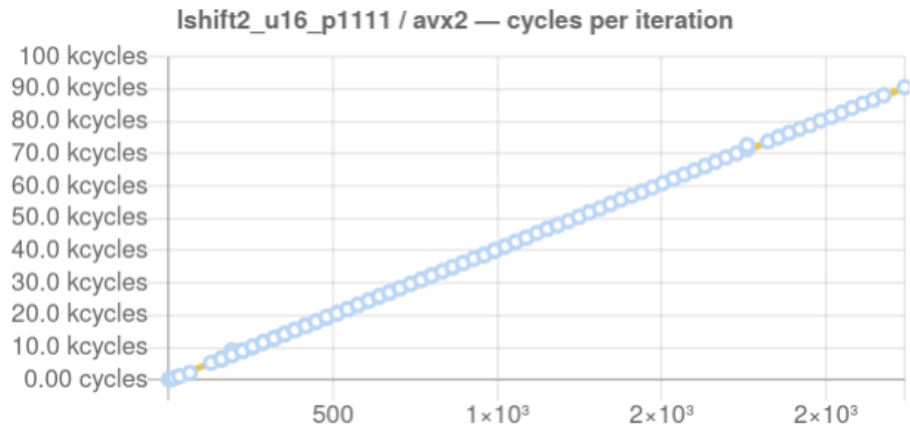
x86 backend


clear_pattern_u16[8]_c
clear_pattern_u16[8]_sse4
clear_pattern_u16[8]_avx2
clear_pattern_u16[9]_c
clear_pattern_u16[9]_sse4
clear_pattern_u16[9]_avx2
clear_pattern_u16[10]_c
clear_pattern_u16[10]_sse4
clear_pattern_u16[10]_avx2
clear_pattern_u16[11]_c
clear_pattern_u16[11]_sse4
clear_pattern_u16[11]_avx2
clear_pattern_u16[12]_c
clear_pattern_u16[12]_sse4
clear_pattern_u16[12]_avx2
clear_pattern_u16[13]_c
clear_pattern_u16[13]_sse4
clear_pattern_u16[13]_avx2

x86 backend

lshift2_u16_p1111_avx2





	lower bound	**estimate**	upper bound
Adjusted cycles	36.9 cycles	37.7 cycles	38.6 cycles
Adjusted time	5.69 ns	8.35 ns	13.5 ns
Raw cycles	39.6 cycles	40.4 cycles	41.2 cycles
Raw time	6.10 ns	8.94 ns	14.4 ns
Ratio (vs ref)	5.54	5.71	5.88

5.71x

libswscale reimagined

Future direction

Future direction

- **Runtime SIMD generation**
  - ARM NEON, asmjit
  - Ramiro Polla <ramiro.polla@gmail.com>
- **GPU backend**
  - SwsOps → SPIR-V → Vulkan
- **Missing functionality**
  - Scaling, LUTs, subsampling, Bayer, EOTF/OETF, ...

How to help

- **Moar SIMD**
  - NEON, RISC-V, ...
- **Testing**
  - SWS_UNSTABLE (*-sws_flags unstable*)
  - ./configure --enable-unstable
- **Funding**
  - (your name could be here)

The End

Thank you for listening!

contact@niklashaas.de