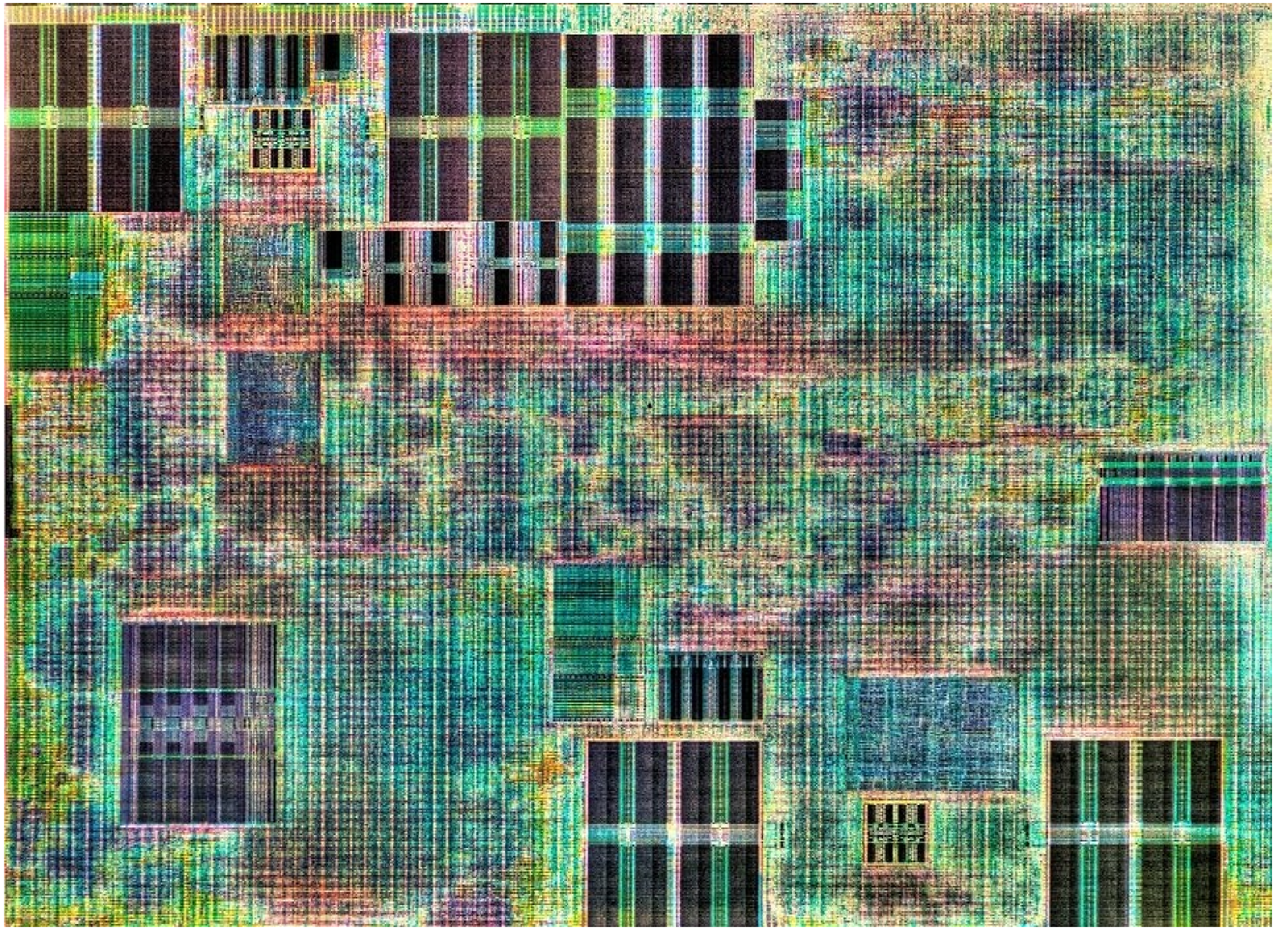
NEON Intrinsics

Michael Hope, Toolchain

bzr branch lp:~michaelh1/+junk/intrinsics-demo

What's NEON?



- Ch 19 'Introducting NEON'

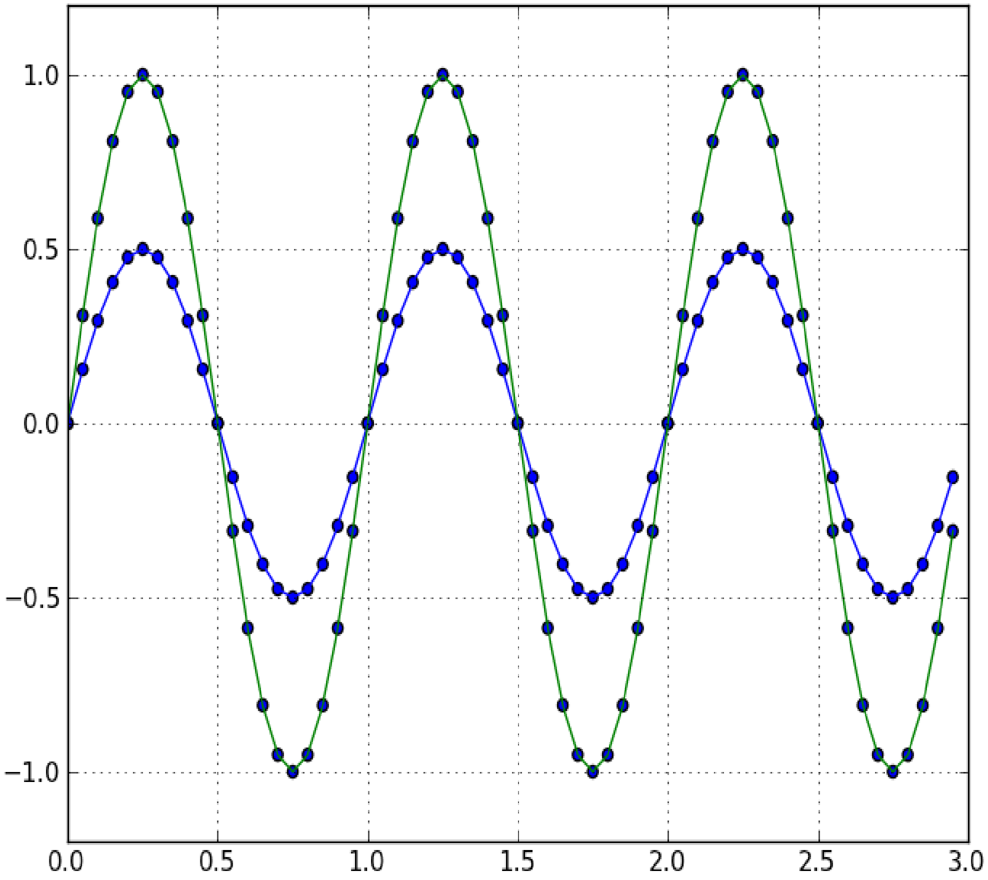
http://infocenter.arm.com/help/topic/com.arm.doc.den0013a/

SIMD is...

Same instruction, many values

Anything involving signals is great for SIMD

Normalisation



Advantages

- Easier to read and write
- Easier (better?) register allocation
- Compiler knows how to schedule
- ABI neutral

Works across compilers

```
> gcc-mcpu=cortex-a9 -mfpu=neon -O3 -c test.c

> armcc --cpu Cortex-A9 --c99 -O3 -c test.c

> clang -mcpu=cortex-a9 -mfpu=neon -O3 -c test.c
```

Tune for the architecture

```
-mtune=cortex-a9

-mtune=cortex-a8

-mtune=cortex-a5
```

SMS, unrolling, profiling?

Writing

Environment

```
#include <arm_neon.h>
```

```
gcc -march=armv7-a -mfpu=neon
```

Data types

<type>x<lanes>_t (`uint8x4_t`)

<type>x<lanes>x<# registers>_t (`int16x2x4_t`)

Some Instructions

Add

```
uint16x4_t vadd_u16 (
    uint16x4_t left,
    uint16x4_t right
    )
```

Multiply

```
uint64x2_t vmlal_u32
    (uint64x2_t,
     uint32x2_t, uint32x2_t)

int32x4_t vqdmlal_s16
    (int32x4_t,
     int16x4_t, int16x4_t)
```

Strided load

```
uint8x8x2_t vld2_u8
    (const uint8_t *)
```

Form of expected instruction(s):

```
    vld2.8 {d0, d1}, [r0]
```

Documentation

GCC

http://gcc.gnu.org/onlinedocs/gcc/ARM-NEON-Intrinsics.html

ARM

http://infocenter.arm.com/help/topic/com.arm.doc.den0013a

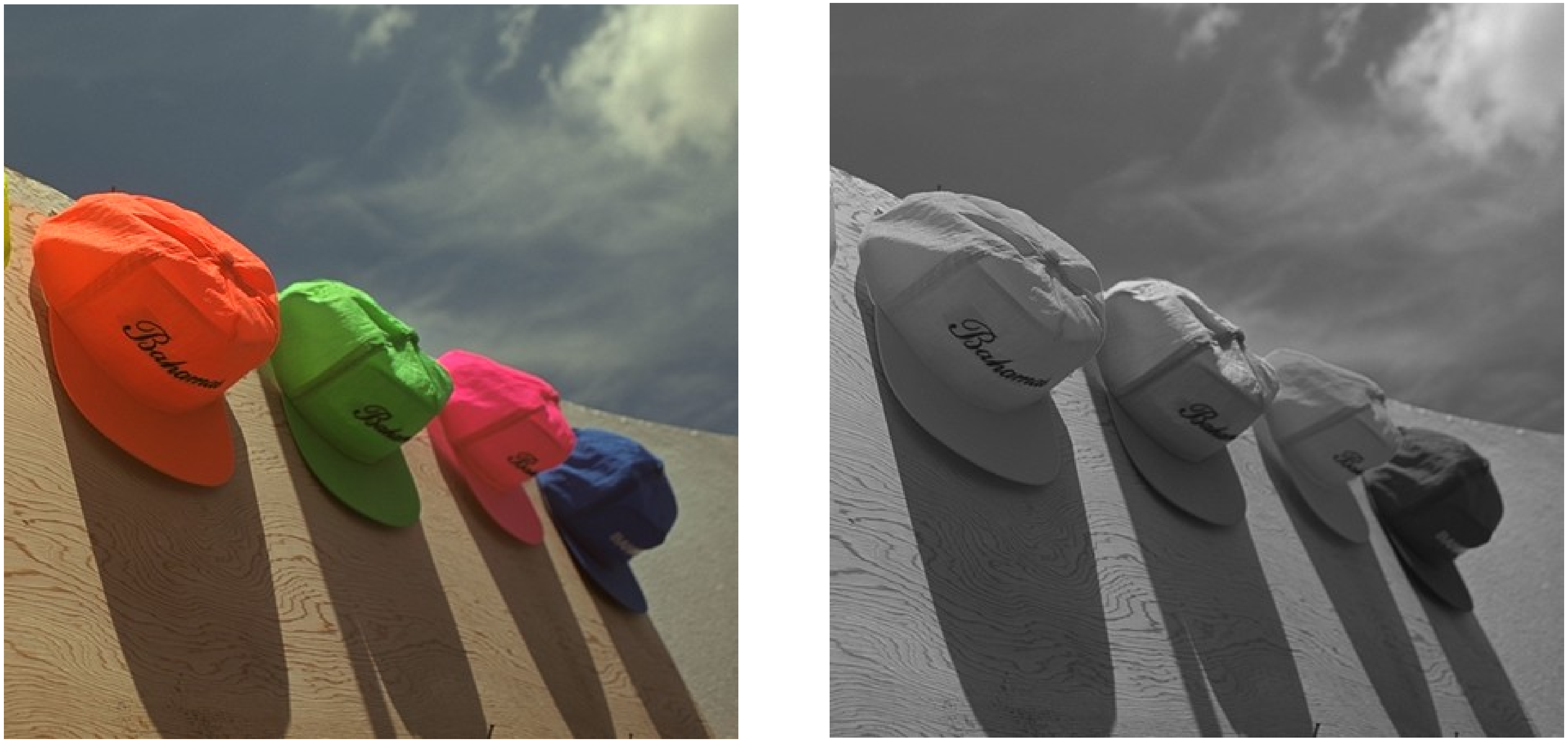
Blog posts

Search for “Coding with NEON” on

http://blogs.arm.com

Writing

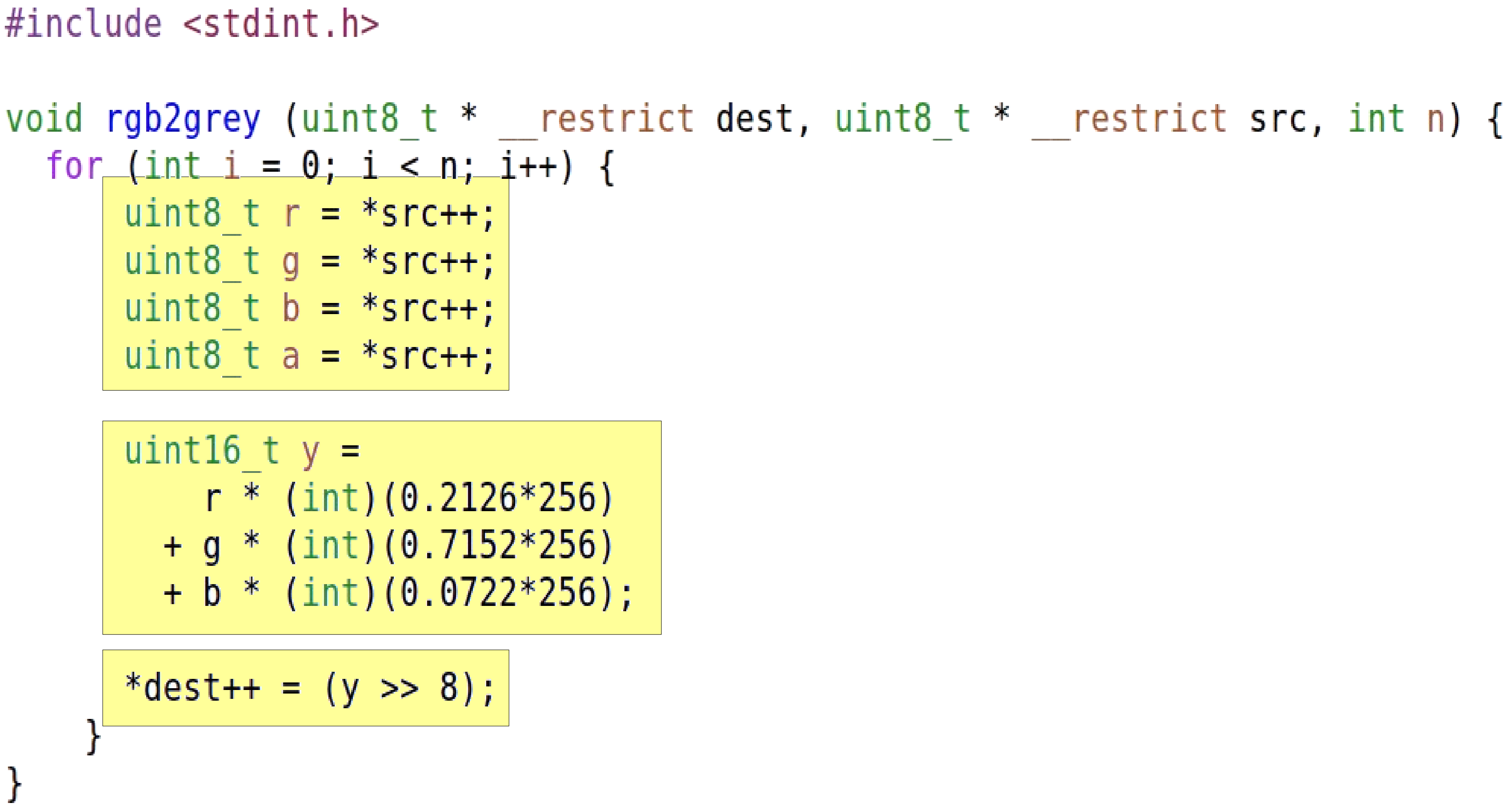
Colour space conversion



$$Y = 0.2126\ R + 0.7152\ G + 0.0722\ B$$

HD television (ITU BT.709)

Versions



```
#include <stdint.h>

void rgb2grey (uint8_t * __restrict dest, uint8_t * __restrict src, int n) {
  for (int i = 0; i < n; i++) {
      uint8_t r = *src++;
      uint8_t g = *src++;
      uint8_t b = *src++;
      uint8_t a = *src++;

      uint16_t y =
          r * (int)(0.2126*256)
        + g * (int)(0.7152*256)
        + b * (int)(0.0722*256);

      *dest++ = (y >> 8);
    }
}
```

Nils Pipenbrinck
http://hilbert-space.de/?p=22

```
        .globl rgb2grey
rgb2grey:
        lsr             r2, r2, #3

        mov             r3, #77
        vdup.8          d4, r3
        mov             r3, #151
        vdup.8          d5, r3
        mov             r3, #28
        vdup.8          d6, r3

.loop:
        vld4.8          {d0-d3}, [r1]!

        vmull.u8        q8, d0, d4
        vmlal.u8        q8, d1, d5
        vmlal.u8        q8, d2, d6

        vshrn.u16       d7, q8, #8
        vst1.8          {d7}, [r0]!

        subs            r2, r2, #1
        bne             .loop

        bx                          lr
```

```
#include <stdint.h>
#include <arm_neon.h>

#define FACTOR  8
#define WIDTH   4

void rgb2grey (uint8_t * __restrict dest, uint8_t * __restrict src, int n) {
 uint8x8_t rcoeff = vdup_n_u8(0.2126*256);
 uint8x8_t gcoeff = vdup_n_u8(0.7152*256);
 uint8x8_t bcoeff = vdup_n_u8(0.0722*256);

 for (int i = 0; i < n; i += FACTOR) {
   uint8x8x4_t rgba = vld4_u8(src);
   uint16x8_t acc;

   acc = vmull_u8(rgba.val[0], rcoeff);
   acc = vmlal_u8(acc, rgba.val[1], gcoeff);
   acc = vmlal_u8(acc, rgba.val[2], bcoeff);

   uint8x8_t result = vshrn_n_u16(acc, 8);
   vst1_u8(dest, result);

   src += FACTOR*WIDTH;
   dest += FACTOR;
  }
}
```

```
rgb2grey:
        cmp     r2, #0
        vmov.i8 d24, #54
        vmov.i8 d23, #183
        vmov.i8 d22, #18
        ble     .L1
        subs    r3, r2, #1
        add     r2, r0, #8
        lsrs    r3, r3, #3
        adds    r3, r3, #1
.L3:
        vld4.8  {d18-d21}, [r1]!
        subs    r3, r3, #1
        vmull.u8        q8, d18, d24
        vmlal.u8        q8, d19, d23
        vmlal.u8        q8, d20, d22
        vshrn.i16       d16, q8, #8
        vst1.8  {d16}, [r0]
        mov     r0, r2
        add     r2, r2, #8
        bne     .L3

        bx      lr
```

Performance

Plain C

48.481 s

Assembly

8.727 s (5.55 x faster)

Intrinsics

8.728 s (5.55 x faster)

Bigger Routines

“libpixelflinger: Add ARM NEON optimized scanline_t32cb16”

http://wiki.linaro.org/RichardSandiford/Sandbox/IntrinsicsPerformance

Hand-written

2.831 s

Intrinsics

2.637 s (7.4 % faster)

Linaro CONNECT