Orc

David Schleef
Entropy Wave Inc

What is Orc

A system for describing low-level computation on modern CPUs

Motivation

Motivation

- Want maintainable assembly code

Motivation

- Want maintainable assembly code
- Want to quickly write assembly code

Motivation

- Want maintainable assembly code
- Want to quickly write assembly code
- Want to verify correct behavior

Possible Solutions

- Hand-written assembly
- perfect C compiler
- C with intrinsics
- C with #pragmas (TI C6x, OpenMP)
- Enhanced C (CUDA, GLSL, OpenCL)
- LLVM
- other...

Combinatoric Problem

Video Format Conversion:

23 input formats
23 output formats
9 algorithms
= 4761 functions

Schroedinger motion compensation: 32768 functions
Pixman rendering: >= 1e9 functions

Conclusion: runtime code generation

Orc Parts

- Language for describing computation
- Compiler for language (orcc)

  to intermediate form

  or to SSE/MMX/C/Neon/etc.
- Orc library (liborc-0.4.so)

  Generate and compile functions at runtime

Orc Features

- Active Backends: SSE, MMX, Neon, Altivec, C
- Experimental: C64x, Arm
- Can generate for different CPU microarchitectures
- 194 opcodes
- 8/16/32/64-bit signed/unsigned int
- 32/64-bit float
- 1D, 2D arrays, constant or variable size

Orc Features

- Easy to make Orc optional
- Embedded friendly
-

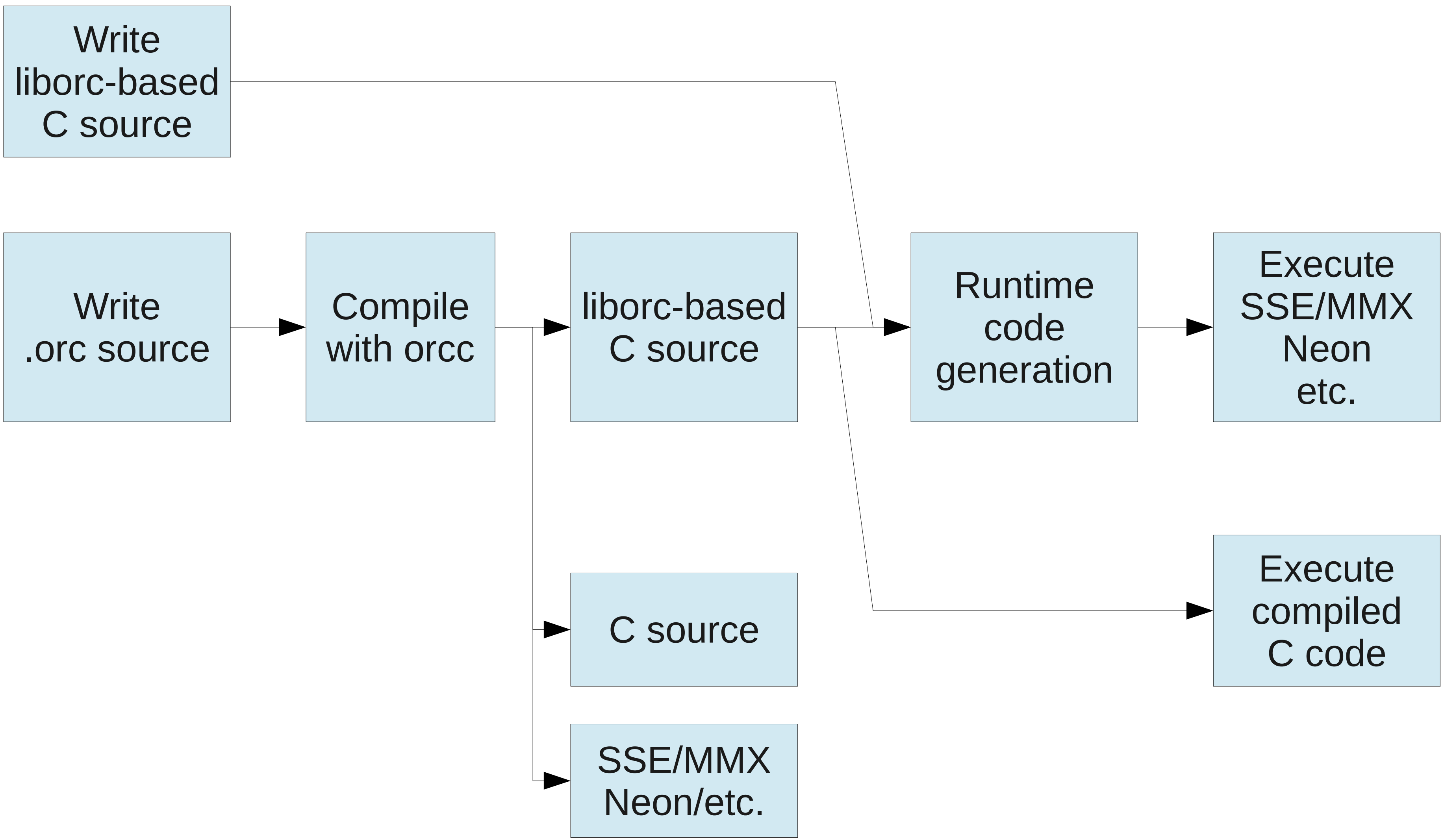
Opcodes

- standard and saturated arithmetic
- shifting, size and float conversion
- specialized loading: loadoff[bwl], ldreslin[bl]
- accumulation
- div255w: divide by 255 (for compositing)
- divluw: divide 16-bit by 8-bit

Automatic Test Features

- Test and compare
  - against backup C code or emulation
- Compile and compare
  - generated source vs. generated binary code

Orc Workflow



Orc code

Vertical downscale by factor of 2 (3 taps)

```
.function cogorc_downsample_vert_cosite_3tap
.dest 1 d1
.source 1 s1
.source 1 s2
.source 1 s3
.temp 2 t1
.temp 2 t2
.temp 2 t3

convubw t1, s1
convubw t2, s2
convubw t3, s3
mullw t2, t2, 2
addw t1, t1, t3
addw t1, t1, t2
addw t1, t1, 2
shrsw t1, t1, 2
convsuswb d1, t1
```

Generated code

Header:

```
void cogorc_downsample_vert_cosite_3tap (uint8_t * d1, uint8_t * s1,
uint8_t * s2, uint8_t * s3, int n);
```

C source (generator function):

```
void
cogorc_downsample_vert_cosite_3tap (uint8_t * d1, uint8_t * s1, uint8_t *
s2, uint8_t * s3, int n)
{
  OrcExecutor _ex, *ex = &_ex;
  static int p_inited = 0;
  static OrcProgram *p = 0;

  if (!p_inited) {
    orc_once_mutex_lock ();
...
}
```

Generated code

C source (backup function):

```
void
static void
_backup_cogorc_downsample_vert_cosite_3tap (OrcExecutor *ex)
{
  int i;
  int8_t * var0;
  const int8_t * var4;
  const int8_t * var5;
  const int8_t * var6;
...
}
```

Test Code: 110 lines of C code

Assembly Code (optional): 395 for SSE, 216 for Neon

GStreamer Plugins using Orc

adder

audioconvert

videoscale

videotestsrc

volume

deinterlace

videobox

videomixer

cog

colorspace

invtelecine

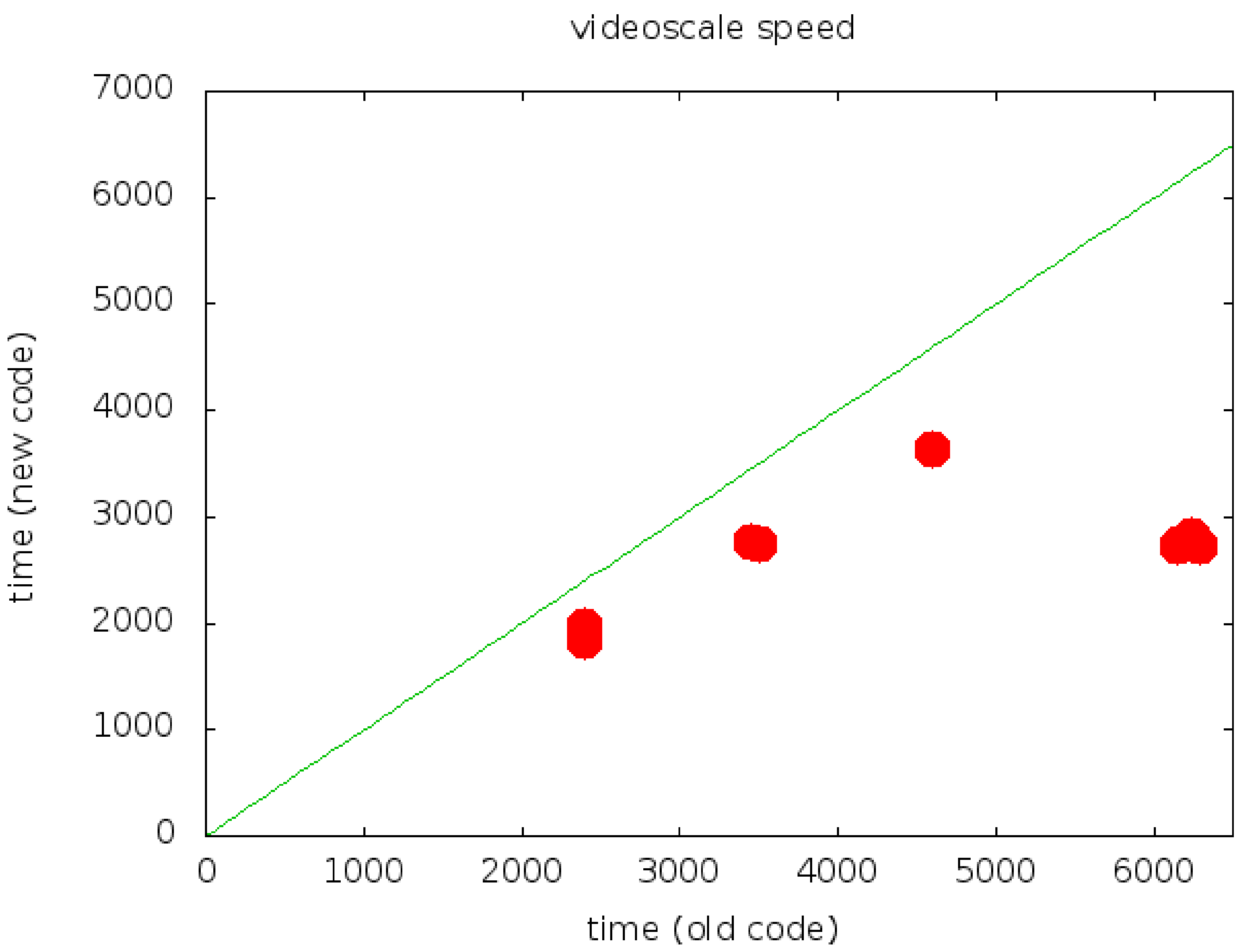
Schrödinger Orc status

- Used everywhere in schro
- Limited by Orc features

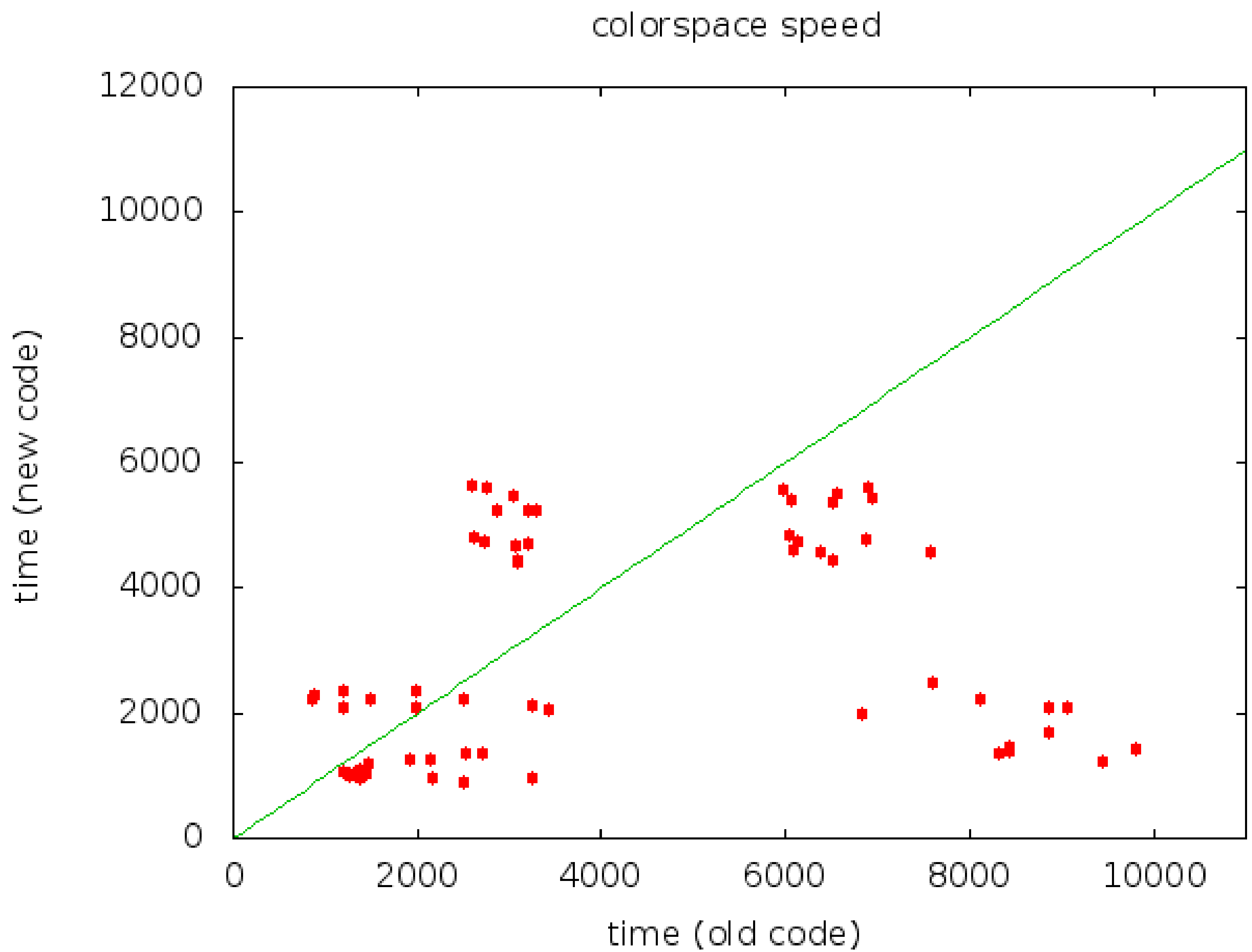
Cairo Orc status

- Orc backend is slightly faster than SSE
- Orc backend handles more operators than SSE backend
- Everything in place to write a Grand Unified Compositor function (>1e9 combinations)

videoscale speed comparison



colorspace speed comparison



Emergent Features

What opportunities arise when writing SIMD code is quick and easy?

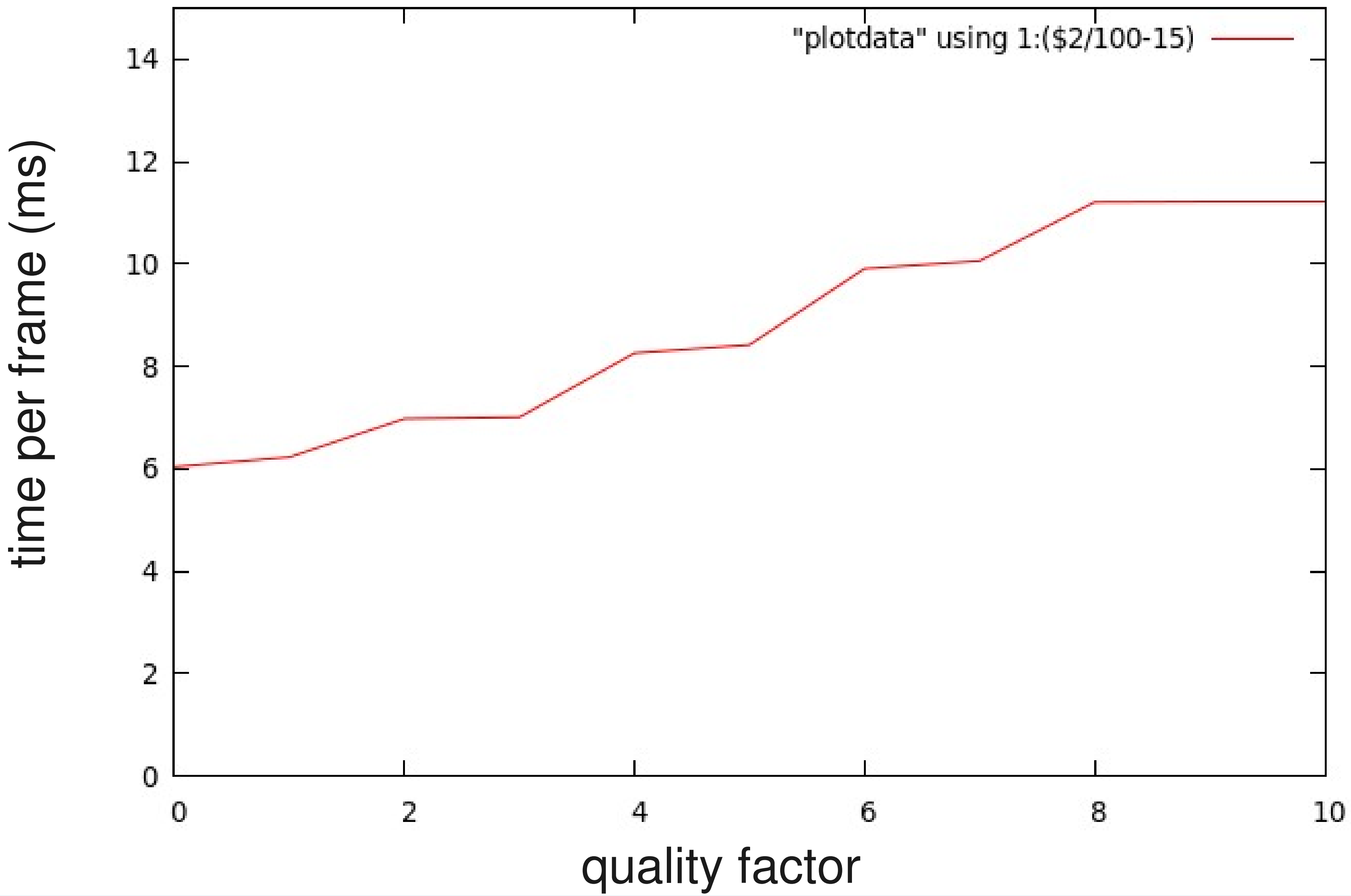
Emergent Features

10/16-bit video processing

floating point video processing

quality vs. time tradeoffs

Emergent Features



Limitations

- 0.4 ABI is horrific
- Fixed-size arrays everywhere
- Limited number of constants/parameters

Opportunities

- Instruction Scheduler

  Reorder instruction stream to improve processor parallelization

- Multi-register allocation

  Do more operations on full registers

- Better handling of register spills/constant loading

Future Directions

ENTROPYWAVE
VIDEO COMPRESSION TECHNOLOGY

- Alignment characteristics for arrays
- Swizzling, shuffling opcodes
- Table lookup opcodes
- Convolution load opcodes
- Non-loop-based functions (for 8x8 DCT)
- Exposure of backend code generators in API
- Macros/high-level opcodes