GOOGLE SUMMER OF CODE 2023

x86_64 Assembly code for VVC decoding filters

Contributor: Shaun Loo
Mentor: Nuo Mi

April 1, 2023

1 ABSTRACT

Versatile Video Codec (a.k.a VVC, H.266) is the successor to the popular High Efficiency Video Codec (a.k.a HEVC, H.265) video compression standard. In anticipation of industry adoption, the FFmpeg community has embarked on the ffvvc project, aiming to bring a VVC decoder into FFmpeg, the open source multimedia processing framework. FFmpeg is the backbone to many open source multimedia technologies, so a high-performance VVC decoder in FFmpeg will allow many other projects to process VVC-encoded video without the baggage of closed-source decoders.

While modern compilers like Clang and GCC can employ tons of optimization techniques to speed up a compiled program, compilers are still not capible of generating high-performance code for DSP and video processing workloads. Modern compilers also struggle with the SIMD extensions introduced by CPU manufacturers, such as the AVX2 extensions released by Intel in 2014. These SIMD extensions massively improve performance in DSP and video processing workloads, and FFmpeg's HEVC decoder already employs AVX2 SIMD instructions to massively improve performance.

The ffvvc project has a C implementation of a vvc decoder. However, its performance leaves much to be desired. This proposal aims to speed up the ffvvc decoder by introducing an `x86_64` implementation of one of the filters used in VVC decoding, utilizing SIMD extensions such as AVX2 to attain performance improvements.

2 IMPLEMENTATION PLAN

This project consists of three components:

2.1 SAO Edge Filter

In applying for FFmpeg's GSoC 2023 program, I had to conduct a "qualification task" which involved re-using existing assembly code from the HEVC SAO filter into the ffvvc decoder. The SAO filter is comprised of two filters - a band filter and an edge filter.

While the band filter was successfully moved and produced the correct output, the edge filter proved to be slightly more difficult, and I was not able to debug the edge filter's erratic result in time for the GSoC deadline.

I plan to work on the SAO edge filter, hopefully being able to complete the SAO filter during the GSoC 2023 early into the program, if not earlier.

2.2 Deblocking filter

A deblocking filter is typically applied to the decoded video output to improve visual quality. Indeed, I, along with most people, will associate blocky video output with inferior visual quality. Hence, a deblocking filter is necessary to attain a visually pleasing output.

Currently, a C version of the VVC deblocking filter exists in the ffvvc tree, found in `libavcodec/vvc_filter.c`). I plan to study this implementation of the deblocking filter closely and create an `x86_64` assembly implementation, once again using the SIMD extensions to acheive a speedup.

2.3 `checkasm`

`checkasm` is FFmpeg's testing framework for testing assembly code and compairing its output to its C counterpart. It is crucial for any new assembly code added to FFmpeg to have tests in `checkasm`. I plan for all accompanying new code to have its own test in the `checkasm` suite.

3 Deliverables

3.1 List of deliverables

- VVC SAO Filter
  - Edge filter
    - Fix all bugs that cause `checkasm` to fail
- VVC Deblocking Filter
  - An `x86_64` implementation of the deblocking filter
  - `checkasm` tests for VVC Deblocking filter
  - A `x86_64` VVC deblocking filter that performs identically to the C version, but faster

4 Timeline

This week-by-week timeline provides a rough idea of my plan to deliver in GSoC 2023.

4.1 April 5 - May 4 (Before the Community Bonding Period)

- VVC SAO Filter
  - Work on bringing over the accelerated SAO edge filter from HEVC

- Communication
  - Continue communicating with Muo Mi over email, as well as engaging the FFmpeg community in the `ffmpeg-devel` IRC channel on `libera.chat`

4.2 May 4 - May 29 (Community Bonding Period)

- VVC SAO Filter
  - Finalizing work on bringing over the accelerated SAO edge filter from HEVC, passing all `checkasm` tests
- VVC Deblocking Filter
  - Study the C implemention of the VVC deblocking filter
  - Study the VVC deblocking filter specifications found in the ITU-T H.266 document.
- Communication
  - Continue communicating with Nuo Mi over email, as well as engaging the FFmpeg community in the `ffmpeg-devel` IRC channel on `libera.chat`
- Other
  - Set up at least two different development environments with different levels of `x86_64` SIMD extension support.

4.3 May 29 - June 18 (Coding Period Begins)

- VVC SAO Filter
  - Finalizing work on bringing over the accelerated SAO edge filter from HEVC (if needed)
- VVC Deblocking Filter
  - Re-implementation of at least two of the deblocking functions in `x86_64` assembly
  - Start work on writing `checkasm` tests for all function that will be written

4.4 June 18 - July 14 (Mid term evaluation)

- VVC Deblocking Filter
  - Finalize at least two implementations of the functions that comprise the full deblocking filter, must pass `checkasm` fully
- Finalize work on `checkasm` tests for all functions that will be written

4.5 July 14 MID TERM EVALUATION DELIVERABLES

- VVC SAO Edge Filter
  - Complete VVC SAO Edge Filter implementation
- VVC Deblocking Filter
  - Full `x86_64` assembly implementation of two of the functions that comprise the VVC deblocking filter
    - These must pass the `checkasm` tests
  - `checkasm` tests for all functions that will be written

4.6 July 14 - July 31

- VVC Deblocking Filter
  - Work on the rest of the functions that comprise the full deblocking filter, must pass `checkasm` fully

4.7 July 31 - August 15

- VVC Deblocking Filter
  - Finalize work on implementing the functions that comprise the full deblocking filter, must pass `checkasm` fully
    - Most of the functions should pass `checkasm` in this period.
- Documentation
  - Documentation of code should happen here, if time permits and documentation is desired.

A buffer of one week is kept for any unforseen delays that may occur.

5 About Me

5.1 Personal Background

I am a junior (third-year) Computer Science student at the University of Minnesota. Even before deciding on my major, I already had a keen interest in video codecs and ran FFmpeg on my computers (almost daily during the pandemic) to produce re-encodes of my favouite videos, figuring out whatever codec and settings produced the best quality for whatever device I was targeting, whether it be DVD, XViD on old computers, or my iPad with 32GB of storage.

My current study focus is on high-performance computer systems. One school assignment I remember fondly is an optimization problem, and a 11x speedup of a 1024×1024 matrix transpose-and-multiply was needed for full credit. With SSE4.2 SIMD extensions, I was able to pull off a 35x speedup.

5.2 How did I hear about this program?

My friend participated in FreeBSD's GSoC Program last year, and I found out that FFmpeg was part of GSoC while browsing for projects to work on this year.

5.3 Summer plans

If accepted, GSoC will be my priority for Summer 2023. I currently have no plans for the summer, and will be able to sustain at least 40 hours a week, if not more.

Summer break starts 10 May 2023, and my availability will be limited 4-9 May 2023 as I finish up finals.