ОПТИМИЗАЦИЯ АППАРАТНО-ПРОГРАММНОГО СРЕДСТВА СШИВКИ ИЗОБРАЖЕНИЙ

Оверченко А. С.
Факультет компьютерных систем и сетей, Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: alex.overchenko.bsuir@gmail.com

Данная статья описывает улучшения времени работы для разрабатываемого аппаратно-программного средства сшивки изображений. Целевой платформой разработки является DSP-процессор ADSP-BF609 от Analog Devices, управляемый операционной системой uClinux. Сборка тестового файла проводилась при помощи компилятора bfin-uclinux-gcc версии 4.3.5.

Введение

В эпоху наступления цифровой эпохи все чаще и чаще возникает вопрос проверки подлинности пользователей. При живом общении основным способом проверки факта, что человек является тем, за кого себя выдает, является верификация его документа, удостоверяющего личность. Такие платежные системы, как WebMoney и Яндекс.Деньги, используют личные встречи своих представителей с проверяемым человеком для проверки документов на их истинность.

I. Разрабатываемое устройство

Для ускорения процесса проверки подлинности паспортов разрабатывается отечественный аналог аппаратно-программного устройства сшивки изображений с расширенным световым диапазоном. Под расширенным световым диапазоном понимается возможность получения устройством изображений как в видимых спектрах света: красном, синем, зеленом – так и в обычно невидимых: инфракрасном и ультрафиолетовом.

Получение изображения в ультрафиолетовом спектре света позволяет увидеть специально нанесенные пометки, наличие которых является свидетельством подлинности верифицируемого документа. Пример изображения паспорта Республики Беларусь с видимым ультрафиолетовым диапазоном приведен на рисунке 1 [1].

Рис. 1 – Паспорт РБ под ультрафиолетовой лампой

По требованию заказчика в устройстве отсутствуют механизмы, отвечающие за движение, что привело к необходимости добавления в устройство двух дополнительных камер, на основании данных с которых и производится обнаружение движения сканируемого объекта. Этот факт послужил причиной необходимости синхронизации данных от основного источника сканирования и камер, отвечающих за получение информации о движении. Как следствие, в устройство была добавлена ПЛИС фирмы Xilinx, с помощью которой обеспечивается настройка устройств получения данных и группирование отсканированной информации для дальнейшей ее передачи на основное устройство управления.

В качестве основного обрабатывающего устройства используется DSP-процессор ADSP-BF609, разработанный фирмой Analog Devices. Основное преимущество использования этого рода процессоров заключается в том, что они предназначены для цифровой обработки сигналов (ЦОС), откуда и получили свое название. Данное преимущество обычно реализуется за счет добавления специфического набора команд. Например команда умножения с накоплением (multiply and accumulate, MAC) используется как основная в большинстве алгоритмов ЦОС и обычно реализуется на соответствующих процессорах за один такт [2].

В качестве операционной системы (ОС) на процессоре используется uClinux [3]. Это открытый проект, который является результатом портирования ОС Linux на микроконтроллеры и микропроцессоры, которые не имеют блока управления памятью (memory management unit, MMU). Он позволяет использовать POSIX-совместимые вызовы системных процедур, предоставляет доступ к сетевыми интерфейсам и виртуальной файловой системе для взаимодействия с внешней и внутренней памятью устройства. Наличие совместимых с Linux программных интерфейсов позволяет ускорить разработку встраиваемого программного обеспечения как с помощью сборки и запуска тестируемого приложения на персональном компьютере, так и на конечном устройстве без необходимости покупки дорогостоящих отладочных средств для

взаимодействия с процессором без операционной системы.

Таким образом, в нашей работе будет произведен анализ различных улучшений уже реализованных алгоритмов с различными оптимизациями компилятора.

II. Оптимизация работы кода

После разработки конечного устройства и выбора необходимых алгоритмов для получения корректного результата необходимо проводить их оптимизацию для улучшения конкурентных характеристик разработанного устройства. При идентичном качестве выходных данных пользователя начинает волновать такие параметры работы устройства как время получения ответа и время автономной работы.

Для улучшения этих показателей необходимо увеличивать производительность алгоритмов. Сделать это можно различными способами. Мы рассмотрим лишь некоторые из них:

- удаление операций, результаты которых не используются;
- уменьшение количества операций выделения и освобождения памяти;
- ручная реализация функций для блоков фиксированной длины.

Указанные выше улучшения производятся для алгоритма обнаружения движения сканируемого документа. В качестве входных данных использовался один кадр из ПЛИС, который содержит информацию для всех световых каналов. Размер изображения с каждой камеры составлял 32x32 пикселя. Анализ движения проводился на основе данных только с одной камеры.

Для сравнения было взято несколько различных вариантов оптимизаций, поддерживаемых компилятором:

- оптимизация по размеру (Os);
- оптимизация уровня 2 (O2);
- оптимизация уровня 3 (O3).

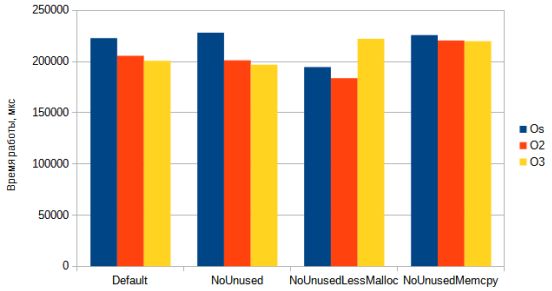

Рис. 2 – Применение ручных оптимизаций для алгоритма обнаружения движения

На рисунке 2 в качестве групп показаны следующие тестируемые варианты (слева направо):

1. изначальный код;
2. удаление неиспользуемых строк кода;
3. удаление неиспользуемых строк кода и уменьшение количество выделений памяти;
4. удаление неиспользуемых строк кода и замена циклов копирования данных на вызовы memcpy библиотеки libc.

Из полученных результатов следует, что различные уровни оптимизации компилятора дают разные значения времени исполнения одиночного обнаружения движения в кадре. В общем случае наблюдается тенденция уменьшения времени работы программы при добавлении оптимизации. Явное использование memcpy вместо циклов показывает увеличение времени работы, что также следует из выдвинутой гипотезы.

Исключение составляет вариант 3 с оптимизацией O3, в котором, предположительно, возникает проблема с кэшем и из-за увеличения размера программы она уже не помещается туда и происходит слишком большое число промахов. Вторым вариантом является то, что адреса объявленных общих массивов не были выровнены, что привело к использованию более медленных команд доступа к памяти или увеличению числа операций обращения к ней.

Заключение

В данной статье описаны базовые улучшения кода, которые позволяют получить небольшой прирост в производительности обычно ценой увеличения размера итогового файла. Они подходят для большинства программ, которые готовы пожертвовать одним в сторону другого, но могут не принести желаемого эффекта в приросте производительности.

Поэтому дальнейшим направлением этой статьи является анализ использования таких средств ускорения как OpenMP, специализированных библиотек под DSP-процессоры, обычно написанные производителем, а также различных способов снижения вычислительной сложности алгоритма.

1. Как делают беларуские паспорта и долго ли их печатают? [Электронный ресурс] / Natatnik – Брест, 2017. – Режим доступа: https://natatnik.by/pasporta/. – Дата доступа: 15.10.2020.
2. Angoletta, M. E. Digital signal processor fundamentals and system design / M. E. Angoletta – Mode of access: https://core.ac.uk/download/pdf/44195315.pdf. – Date of access:: 17.10.2020.
3. uClinux – Embedded Linux Microcontroller Project – Mode of access: http://www.uclinux.org/ – Date of access: 13.11.2018.