Научная визуализация, 2024, том 16, номер 4, страницы 102 - 108, DOI: 10.26583/sv.16.4.09

Разработка и применение алгоритмов фильтрации и обработки видеопотока на ПЛИС

Ю.А. Салькова[1,B], О.Н. Третьякова[2,A], Д.Н. Тужилина[3,B]

[A] ФГБОУ ВО Московский авиационный институт
[B] ООО «Лаборатория промышленных исследований» группы компаний «Лазеры и аппаратура ТМ»

[1] ORCID: 0009-0002-4928-3171, salkov@laser-app.ru
[2] ORCID: 0000-0003-0256-4558, tretiyakova_olga@mail.ru
[3] ORCID: 0000-0002-8570-1732, tuzhilin@laserapr.ru

Аннотация

Статья посвящена деталям реализации алгоритмов машинного зрения на ПЛИС. Алгоритмы были реализованы при помощи инструмента высокоуровневого синтеза Vitis HLS. Главным параметром при реализации алгоритмов является их быстродействие, так как необходимо с минимальной задержкой получать новые данные с приходящего кадра.

Ключевые слова: система на кристалле, ZYNQ, методы машинного зрения, обработка видео, фильтрация изображения на FPGA, технология приборостроения, скользящее среднее, масс-центр, HLS.

1. Введение

В процессе размерной обработки деталей полупроводниковых приборов методом стелс резки необходимо точно фокусировать луч лазера прямо под поверхностью обрабатываемой детали. Для подстройки фокуса лазера прямо в процессе резке была разработана [1] оптическая система слежения (рисунок 1), которая состоит из лазерного диода, линейной камеры и контроллера. Луч лазерного диода под углом светит на поверхность обрабатываемой детали и отражается от нее. Отраженный луч лазерного диода попадает на линейную камеру, тем самым формируя пик на кадре с камеры.

Кадр с камеры поступает в контроллер оптической системы слежения, где сначала проходит фильтрацию фильтром скользящее среднее, после чего на отфильтрованном кадре ищется пик по заданным параметрам, формулой поиска центра масс.

Так как подстройка фокуса силового лазера ведется прямо в процессе резки, контроллеру СОС необходимо быстро принимать и обрабатывать входящий видеопоток, поэтому все алгоритмы машинного зрения и алгоритм регуляции сервомотора были реализованы на мощностях ПЛИС. Реализация алгоритмов на ПЛИС позволяет добиться большей производительности в сравнении с привычными архитектурами вычислительных систем (таких как x86, ARM или RISC-V), так как при реализации алгоритмов на ПЛИС, микросхема становится узкоспециализированным вычислителем для решения конкретных задач.

Рисунок 1. Схема оптической системы слежения

2. Аппаратная конфигурация и средства разработки

Для реализации контроллера системы оптического слежения (СОС) была взята система на кристалле (SOC) от компании Xilinx – отладочная плата AMD Zynq 7000 SoC ZC702. Данная плата является избыточной для реализации контроллера СОС, но она имеет все необходимые интерфейсы и позволяет разрабатывать контроллер, не ограничиваясь количеством логических вентилей.

Для регистрации положения отраженного лазерного пучка была взята линейная монохромная камера, передающая видео по протоколу camera link [2]. Для приема видео потока была куплена специальная интерфейсная плата для ПЛИС с разъемом camera link.

При разработке контроллера СОС был использован инструмент высокоуровневого синтеза vitis-HLS [3]. Инструменты высокоуровневого синтеза позволяют разрабатывать отдельные функциональные блоки при помощи написания кода на высокоуровневых языках (C++, Python и т.д.), а не на языках описания аппаратуры (Verilog, HDL и т.д.). Данные инструменты дают возможность разработчику функционального модуля абстрагироваться от уровня регистровых передач (RTL) и разрабатывать код на уровне работы с данными.

Vitis-HLS имеет адаптированную библиотеку Open-CV, таким образом разработка функциональных блоков, связанных с обработкой видео, становится на много проще, из-за наличия готовых и адаптированных решений. В данной разработке функции open-CV не применялись, из-за специфики работы контроллера СОС.

3. Разработка алгоритма фильтрации

Как уже упоминалось выше, контроллер СОС должен быстро принимать и обрабатывать приходящий видеопоток. Перед поиском пика на кадре, кадр необходимо отфиль-

тровать. Для фильтрации кадра был взят модифицированный алгоритм скользящее среднее [4]. Данный алгоритм был выбран из-за того, что пик на кадре является явно выраженным и необходимо просто убрать лишние шумы, для правильного определения масс центра пика.

На вход фильтру подается изображение разрешением 2*2048 пикселей, кадр является черно-белым, каждый пиксел кодируется 8 битами. Первая строка кадра записывается в буфер кадра без изменений. С начала прихода второй строки кадра начинается фильтрация с помощью скользящего окна, ширина окна задается пользователем, а высота окна фиксирована – 2 пикселя. С начала фильтрации окно заполнено нулями, из-за этого теряется часть информации в начале изображения. Отфильтрованная вторая строка изображения записывается в буфер кадра, таким образом в одном кадре присутствуют, как и “сырые” данные, так и отфильтрованные, за счет этого в визуализирующей программе можно отобразить сразу оба графика. На рисунке 2 изображен алгоритм работы фильтра скользящее среднее.

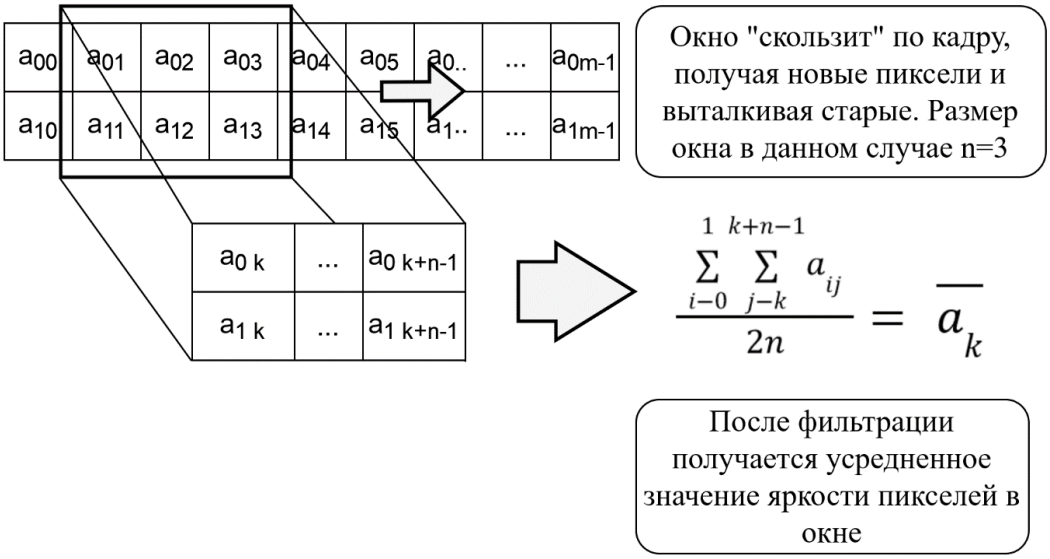

Рисунок 2. Алгоритм и формула фильтрации

На вход фильтра поступает потоковое видео, в каждый такт передается по два пикселя, таким образом при получении второй строки изображения и начале фильтрации кадра в окно записываются сразу по два пикселя с каждой строки.

В качестве буфера кадров выступает блочная память чипа из-за того, что данная память является двухпортовой, невозможно в один момент времени производить операции чтения или записи более с чем двумя ячейками памяти. Это накладывает серьезные ограничения по работе с памятью, так как в такт приходит данные сразу о двух пикселях кадра, и если это уже вторая строка кадра, то в тот же такт необходимо считать данные с буфера, для помещения их в окно усреднения.

Для создания и работы с памятью используется класс библиотеки vitis vision “xf::cv::LineBuffer”. Класс позволяет тонко настроить размеры и тип памяти буфера кадра, а также при создании буфера нужно указать размер элементов, которые будут хранится в нем. Чтобы уложится в ограничение по количеству единовременных операций над буфером было принято решение хранить в одной ячейке буфера сразу по два пикселя, за счет такого решения единовременно производится всего лишь две операции над буфером кадра, а именно операция записи новых пикселей и операция чтения пикселей первой строки кадра.

При фильтрации кадра в окно фильтрации записываются сразу по два пикселя с обоих линий кадра, “вытесняя” старые пиксели. При операции усреднения окна происходит считывание всех ячеек окна, суммирование их и получение среднего. Таким об-

разом применение обычной блочной двухпортовой памяти для окна невозможно, так как будет тратиться слишком много времени на получение данных со всех ячеек окна, поэтому окно усреднения представляет собой набор сдвиговых регистров. Сдвиговые регистры позволяют получить данных со всех ячеек за один такт, а также позволяют произвести операцию сдвига данных, в какую-либо сторону. Регистров в ПЛИС намного меньше, чем блочной памяти, поэтому реализация буфера кадра при помощи регистров особого смысла не имеет.

Окно сдвиговых регистров реализуется при помощи класса “xf::cv::Window”, который также является частью библиотеки vitis vision. При создании окна задаются его высота и ширина, а также размер данных, хранящихся в окне.

Общий алгоритм фильтрации представлен на рисунке 3, на рисунке красным цветом отмечена операция чтения из буфера кадра, а синим цветом отмечены операции записи.

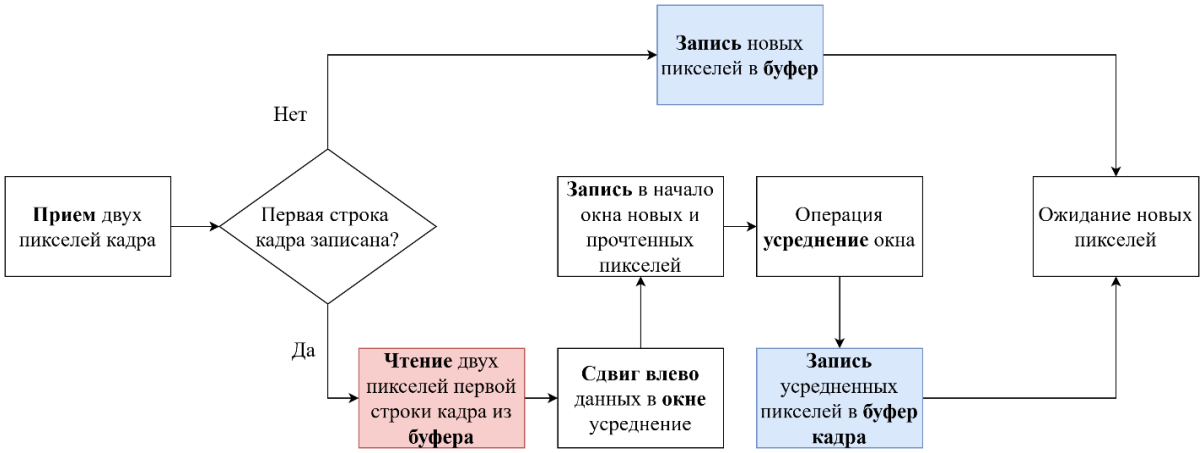

Рисунок 3. Алгоритм фильтрации кадра

При настройке системы важно, чтобы можно было задавать разные размеры сглаживающего окна, это позволит более тонко настроить контроллер. Но при реализации каких-либо алгоритмов на ПЛИС нельзя изменять размеры уже определенных массивов, это связано с тем, что при синтезе и размещении (Place and Route) резервируются все необходимые ресурсы и в дальнейшем нет никакой возможности добавлять или освобождать их [5]. Поэтому для реализации переменного размера окна было решено изначально создавать окно самого максимального размера, чтобы иметь возможность варьировать размер окна от 0 (фильтрации не происходит) до максимального размера.

При операции усреднения окна, итерация по окну происходит при помощи цикла for, в vitis HLS есть специальная прагма для “разворачивания циклов” (pragma HLS unroll) (т.е. выполнения операции в цикле сразу над всеми итерируемыми объектами). Данная прагма позволяет выполнять весь цикл за один такт синхроимпульса, что позволяет значительно ускорить обработку данных. Для “разворачивания” цикла требуется в объявлении цикла указывать константное количество итераций, иначе прагма не применится к циклу. На рисунке 4 представлен пример работы прагмы unroll, здесь maxWindowSize является константной переменной, а windowSize может изменяться пользователем во время работы. За счет приема с if(i<..) в теле цикла, возможно создавать циклы с переменным количеством итераций, которые будут выполняться за один такт синхроимпульса.

```
int avg = 0;

for(int i = 0; i < windowSize; ++i)
{
    avg += window[i];
}

avg /= windowSize;
```

```
int avg = 0;

for(int i = 0; i < maxWindowSize; ++i)
{
    #pragma HLS unroll
    if(i < windowSize) avg += window[i];
}

avg /= windowSize;
```

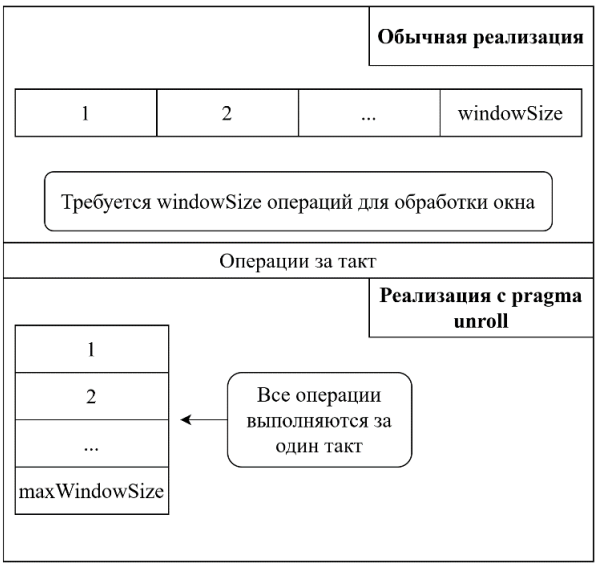

Рисунок 4. Описание работы прагмы unroll

В итоге удалось разработать оптимизированный алгоритм для фильтрации приходящего видеопотока. Фильтр выдает результат сразу как получил полный кадр, за счет этого скорость обработки данных зависит по большей части от скорости передачи видеопотока камерой.

4. Разработка алгоритма поиска пика на кадре

Для поддержания быстродействия контроллера было принято решение реализовать алгоритм поиска пика вместе с алгоритмом фильтрации видеопотока. Таким образом в одном функциональном блоке реализованы два алгоритма, работающие друг за другом. После операции фильтрации окна происходит обновление данных о пике, а сразу после получения всего кадра происходит вычисление нового положения пика. Задержка между получением полного кадра и вычислением нового положения пика является минимально возможной.

Сам алгоритм поиска масс центра пика представляет собой формулу поиска масс центра, где вместо массы была взята яркость конкретного пикселя (m_i), а вместо координат точки был взят индекс пикселя (r_i). Алгоритм поиска масс-центра представлен на рисунке 5.

Поиск пика производится на всех не нулевых отфильтрованных пикселях кадра. Для поиска пика пользователь задает параметры, такие как: максимальная и минимальная ширина пика, порог яркости пикселей пика, максимальная яркость пика и т.д.. Сразу после получения среднего значения с окна усреднения происходит обновление данных, если яркость пикселя равна нулю, но до него пиксели были не нулевые, выполняется расчет масс центра пика и проверка полученных данных о пике с заданными параметрами. Первый пик, удовлетворяющий установленным параметры, считается подходящим и алгоритм заканчивает свою работу и на выход функционального блока поступают данные о положении и размере пика.

При реализации алгоритма поиска пика необходимо сравнивать текущие значение пика с установленными параметрами, это производится при помощи оператора if с большим числом сравнивании (т.е. if (a < b && b == c && ...)). Использование логического И (&& в C++) в одном операторе if в большом количестве, приводит к появлению длинного критического пути, это связано с тем, что в C++ при использовании оператора && гарантируется что сначала проверится условие слева от оператора и, если оно не верно, то второе условие выполнятся не будет. При увеличении критического пути скорость обработки данных сильно снижается, из-за понижения частоты работы всего контроллера. В связи с этим, все операторы логического И были заменены на операто-

ры побитового И (&) при вычислении побитовое И позволяет проверить сразу все условия и только после вычислять результат побитового И.

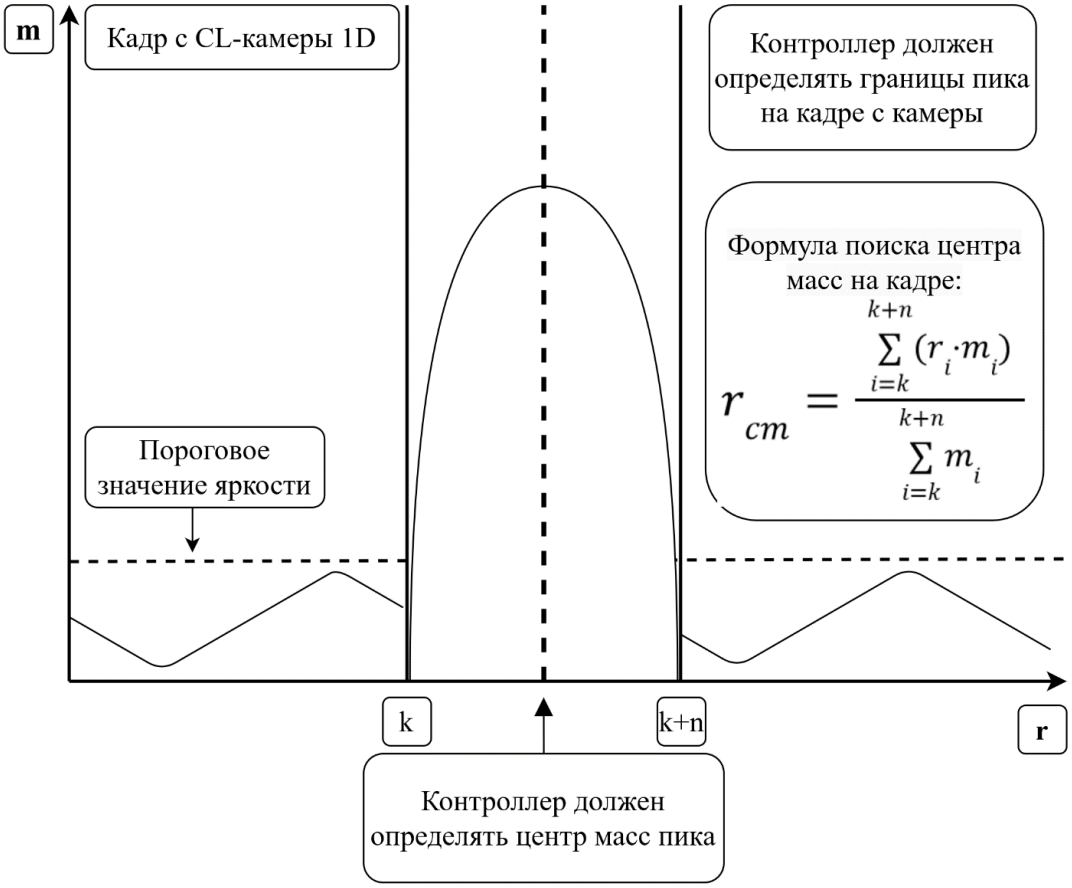

Рисунок 5. Алгоритм поиска масс-центра

В результате разработки алгоритма поиска пика удалось реализовать оптимизированный и высокопроизводительный алгоритм, выдающий результат почти сразу после получения полного кадра. Как и с реализацией алгоритма фильтрации производительность алгоритма также упирается в скорость передачи данных от камеры.

5. Заключение

В результате реализации алгоритмов был разработан функциональный блок фильтрации и поиска пика на видеопотоке. Данный блок обрабатывает приходящий кадр с минимальным временем 42 мкс, данное время включает в себя время получения кадра. Функциональный блок имеет гибкие настройки для фильтрации и поиска пика на кадре.

Благодарности

Авторы выражают благодарность руководству группы компаний «Лазеры и Аппаратура ТМ» за помощь в материально-техническом обеспечении проведения экспериментальных исследований и моделирования рассматриваемого процесса.

Список литературы

1. Кондратенко В.С., Сапрыкин Д.Л., Третьякова О.Н., Тужилин Д.Н. Разработка системы автоматического управления подстройкой фокуса для технологии лазерной микрообработки материалов // Приборы. 2022. № 4. С.26–31.

2. Camera Link standard [Электронный ресурс] // Imagelabs URL: https://www.imagelabs.com/wp-content/uploads/2010/10/CameraLink5.pdf (дата обращения: 07.07.2024)
3. Ryan Kastner, Janarbek Matai, and Stephen Neuendorffer. Parallel Programming for FPGAs [Электронный ресурс] // Kastner Research Group URL: https://kastner.ucsd.edu/wp-ontent/uploads/2018/03/admin/pp4fpgas.pdf (дата обращения: 07.07.2024)
4. Орлова Ирина Анатольевна, Устюков Дмитрий Игоревич, Ефимов Алексей Игоревич Особенности реализации пространственных фильтров изображений на FPGA // Известия ТулГУ. Технические науки. 2018. №2. 195-206.
5. Цифровой синтез: практический курс // Антонов А. А., Барабанов А. В., и др. / Под ред. А. Ю. Романова, Ю. В. Панчула. - М.: ДМК Пресс, 2020. 111.

Scientific Visualization, 2024, volume 16, number 4, pages 102 - 108, DOI: 10.26583/sv.16.4.09

Development and Application of Algorithms of Video Stream Filtering and Processing on Programmable Logic Integrated Circuits FPGA

Yu.A. Salkov[1,B], O.N. Tretiyakova[2,A], D. N. Tuzhilin[3,B]

[A] Moscow Aviation Institute (National Research University) MAI
[B] LLC "PROMIS LAB" of the group of companies "Lasers and Equipment TM"

[1] ORCID: 0009-0002-4928-3171, salkov@laser-app.ru
[2] ORCID: 0000-0003-0256-4558, tretiyakova_olga@mail.ru
[3] ORCID: 0000-0002-8570-1732, tuzhilin@laserapr.ru

Abstract

The paper is devoted to the details of implementation of machine vision algorithms on FPGA. The algorithms were implemented using the Vitis HLS high-level synthesis tool. The main parameter in the realization of the algorithms is speed; since it is necessary to proceed processing of the incoming video stream with minimum delay.

Keywords: system-on-chip, ZYNQ, machine vision techniques, video processing, FPGA image filtering, instrumentation technology, moving average, mass center, HLS.

References

1. Kondratenko, V.S.; Saprykin, D.L.; Tretyakova, O.N.; Tuzhilin, D.N. Development of the automatic control system of the focus adjustment for the technology of laser micromachining of materials (in Russian) // Instrumentation. 2022. № 4. C.26-31.
2. Camera Link standard [Electronic resource] // Imagelabs URL: https://www.imagelabs.com/wp-content/uploads/2010/10/CameraLink5.pdf (date of address: 07.07.2024)
3. Ryan Kastner, Janarbek Matai, and Stephen Neuendorffer. Parallel Programming for FPGAs [Electronic resource] // Kastner Research Group URL: https://kastner.ucsd.edu/wp-ontent/uploads/2018/03/admin/pp4fpgas.pdf (date of address: 07.07.2024)
4. Orlova Irina Anatolievna, Ustyukov Dmitry Igorevich, Efimov Alexey Igorevich Features of realization of spatial image filters on FPGA // Izvestiya TulSU. Technical Sciences. 2018. №2. 195-206.
5. Digital synthesis: a practical course // Antonov A. A., Barabanov A. V., et al. V., Barabanov A., et al. / Edited by A. Y. Romanov, Y. V. Panchul. - Moscow: DMK Press, 2020. 111.