© 2012 г. И.В. ЕГОРОВ, А.А. ВНУКОВ

(Московский институт электроники и математики Национального исследовательского университета «Высшая школа экономики» (МИЭМ НИУ ВШЭ), Москва)

РАЗРАБОТКА ВЫСОКОПРОИЗВОДИТЕЛЬНОГО ПАРАЛЛЕЛЬНОГО АЛГОРИТМА МАСШТАБИРОВАНИЯ ИЗОБРАЖЕНИЙ

В статье рассматриваются вопросы, связанные с масштабированием цифровых изображений, оптимизацией проводимых вычислений путём использования параллельной обработки. В качестве наиболее подходящей структуры планируется применение многоступенчатого конвейера. Внесено предложение использовать современные ПЛИС в качестве аппаратной основы.

DEVELOPMENT OF HIGH-PERFOMANCE PARALLEL SCALING ALGORITM / I.V. Egorov, A.A.Vnukov (Moscow Institute of Electronics and Mathematics of National Research University Higher School of Economics (MIEM HSE), B. Trehsvyatitelsky pereulok 3, Moscow, 109028, Russia). This articlcle deals with digital image scaling and optimizing of calculations by means of parallel processing. It is argued that the most efficient structure for this task is that of many-staged pipeline. It is proposed to use FPGA as the hardware basis.

1. Введение

Общая постановка задачи предполагает реализацию распараллеливания процесса масштабирования цифровых изображений.

Пусть имеется цифровое изображение, полученное от некого источника, размер которого необходимо изменить перед отображением.

Современные алгоритмы масштабирования позволяют качественно и быстро масштабировать изображение, однако за счёт распараллеливания вычислений можно добиться повышения скорости обработки изображений, что особенно важно в критичных ко времени областях, таких как видеообработка.

Таким образом, необходимо выполнить научно-исследовательскую работу по анализу современных алгоритмов и установлению путей оптимизации проводимых вычислений, а также подбору подходящей современной элементной базы для реализации этих идей.

Целью исследования является выполнение следующих работ:

1) рассмотрение представления цифровых изображений в системах отображения информации;
2) рассмотрение общего процесса масштабирования и порождаемых им артефактов
3) анализ современных алгоритмов масштабирования;
4) внесение предложений о возможном распараллеливании вычислений, проводимых алгоритмами масштабирования.

2. Представление цифровых изображений в системах отображения информации

В системах отображения информации применяется растровая графика, как наиболее близкая по формату представления изображения к средствам отображения (дисплеям).

Растровое изображение — изображение, представляющее собой сетку пикселей - или точек цветов (обычно прямоугольную) на компьютерном мониторе, бумаге и других отображающих устройствах и материалах.

Важными характеристиками изображения являются:

1) количество пикселей - разрешение. Может указываться отдельно количество пикселей по ширине и высоте (1024*768, 640*480,...) или же, редко, общее количество пикселей (часто измеряется в мегапикселях);

2) количество используемых цветов или глубина цвета (эти характеристики имеют следующую зависимость: N = 2I, где N - количество цветов, а I - глубина цвета);

3) цветовое пространство (цветовая модель) RGB, CMYK, XYZ, YCbCr и др.

В наше время в системах отображения наиболее широкое применение имеет цветовая модель RGB (аббревиатура английских слов Red, Green, Blue — красный, зелёный, синий) — аддитивная цветовая модель, описывающая способ синтеза цвета для цветовоспроизведения. В российской традиции иногда обозначается как КЗС.

Выбор основных цветов обусловлен особенностями физиологии восприятия цвета сетчаткой человеческого глаза..

Аддитивной она называется потому, что цвета получаются путём добавления (англ. addition) к черному. Иначе говоря, если цвет экрана, освещённого цветным проектором, обозначается в RGB как (r1, g1, b1), а цвет того же экрана, освещенного другим проектором, — (r2, g2, b2), то при освещении двумя проекторами цвет экрана будет обозначаться как (r1+r2, g1+g2, b1+b2).

Изображение в данной цветовой модели состоит из трёх каналов, т.е. каждый пиксель хранит в себе информацию о значении трёх цветовых компонент независимо. При смешении основных цветов (основными цветами считаются красный, зелёный и синий) — например, синего (B) и красного (R), мы получаем пурпурный, при смешении зеленого (G) и красного (R) — жёлтый, при смешении зеленого (G) и синего (B) — цианový. Смешивание основных цветов в разных пропорциях даёт разный цвет. При смешении всех трёх цветовых компонентов мы получаем белый цвет.

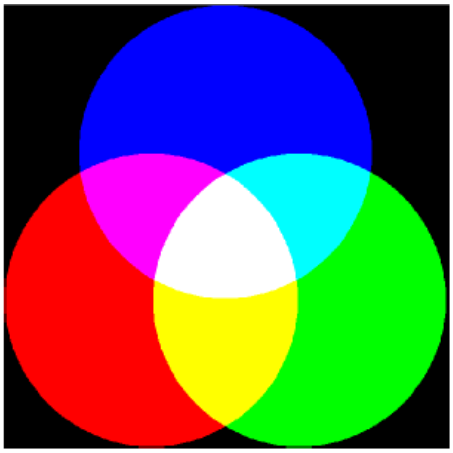

Рис.1. Цветовая схема RGB

3. Общие сведения о масштабировании цифровых изображений

Масштабирование - изменение размера изображения с сохранением пропорций. Под масштабированием подразумевается как увеличение (интерполяцию), так и уменьшение (децимацию) размеров изображения (разрешения изображения). В настоящее время масштабирование выполняется с помощью компьютерной техники. В растровой графике при масштабировании происходит потеря качества изображения. Искажения изображения при масштабировании, приводящие к потере качества, называются артефактами масштабирования. Типичные артефакты при изменении разрешения изображения:

1) пикселизация – распад изображения на квадраты различных цветов, из-за значительного уменьшения разрешения;
2) звон – появление светлых контуров или «теней» вдоль границ объектов;
3) алиасинг или лестничный эффект – появление угловатых контуров вдоль границ объектов («угловатость» изображения);
4) размытие – потеря чёткости изображения.

Выбор алгоритма масштабирования является результатом компромисса между типом и выраженностью артефактов, и вычислительной сложностью преобразования, от которой напрямую зависит время затрачиваемое на обработку изображения. Так как от выбранного алгоритма зависит качество результатов обработки и время, затрачиваемое на обработку изображений, а следовательно и требования к производительности технических средств.

Большая часть современных алгоритмов основывается на предварительном расчёте коэффициентов, обозначающих пропорции в которых будут смешиваться цвета соседних пикселей при масштабировании. Также существует класс алгоритмов, базирующихся на предварительном анализе изображения, для расчёта уникальной для каждого изображения матрицы коэффициентов, как правило подобные алгоритмы выдают на выходе более качественное изображение, но требуют более сложных вычислений, к тому же в режиме реального времени. Примерами алгоритмов первой группы являются:

1) метод ближайшего соседа;
2) билинейная интерполяция;
3) бикубическая интерполяция.

Примером алгоритма второй группы является алгоритм направленной интерполяции.

4. Сравнительный анализ распространённых методов масштабирования

Простейший алгоритм масштабирования предполагает периодическое дублирование или исключение точек изображения. Так, например, для масштаба 99 % или 101 % соответственно убирают или дублируют каждую сотую точку. При таком способе масштабирования утрачиваются или искажаются по толщине тонкие линии и засечки некоторых знаков. Особенно заметны такие искажения при нарушении общего ритма и гармонии композиции, появлении ложных узоров. В целом он обладает низкой эффективностью даже при малых изменениях масштаба, но его вычислительная сложность очень низка, что положительно сказывается на скорости работы. Такой алгоритм называют методом ближайшего соседа или ближайшего пикселя.

Более сложным, но и более качественным методом является метод билинейной интерполяции, названный так по соответствующей математической операции. Применительно к компьютерной графике, в данном методе в качестве значения функции выступает цвет пикселя (его составляющие). При этом квадрат, образованный четырьмя рассматриваемыми основными точками принимается единичным. Билинейная

интерполяция — в вычислительной математике расширение линейной интерполяции для функций двух переменных. Ключевая идея заключается в том, чтобы провести обычную линейную интерполяцию сначала в одном направлении, затем в другом. Допустим, что надо интерполировать значение функции f в точке P = (x, y). Для этого необходимо знать значения функций в точках, окружающих P, т.е. в точках Q11 = (x1, y1), Q12 = (x1, y2), Q21 = (x2, y1), и Q22 = (x2, y2). Первым шагом интерполируется (линейно) значение вспомогательных точек R1 и R2 вдоль оси абсцисс, где R1 = (x,y1), R2 = (x,y2). В таком случае значение функции в точках R1 и R2 вычисляется по следующим формулам:

$$f(R1) \approx \frac{x2 - x}{x2 - x1} f(Q11) + \frac{x - x1}{x2 - x1} f(Q21),$$
$$f(R2) \approx \frac{x2 - x}{x2 - x1} f(Q12) + \frac{x - x1}{x2 - x1} f(Q22)$$

Далее проводится линейная интерполяция между вспомогательными точками R1 и R2, согласно формуле:

$$f(P) \approx \frac{y2 - y}{y2 - y1} f(R1) + \frac{y - y1}{y2 - y1} f(R2)$$

Это и есть приблизительное значение функции в точке P, то есть в раскрытом виде эта формула имеет вид:

$$f(x,y) \approx \frac{f(Q11)}{(x2 - x1)(y2 - y1)}(x2 - x)(y2 - y) + \frac{f(Q21)}{(x2 - x1)(y2 - y1)}(x - x1)(y2 - y) +$$
$$+ \frac{f(Q12)}{(x2 - x1)(y2 - y1)}(x2 - x)(y - y1) + \frac{f(Q22)}{(x2 - x1)(y2 - y1)}(x - x1)(y - y1)$$

Одним из минусов билинейной интерполяции при масштабировании изображений является тот факт, что при увеличении в N раз изображения размером W на H пикселей в результате будет получено изображение размером не NW на NH пикселей, а (N(W − 1) + 1) на (N(H − 1) + 1) пикселей.

Связано это с тем, что в исходном изображении, например, по горизонтали имеется W точек, то есть (W − 1) смежных пар. При выполнении операции увеличения изображения в N раз между каждой парой основных точек вставляется по (N − 1) дополнительных точек (то есть при увеличении вдвое между основными точками вставляется еще по одной, при увеличении втрое - по две и т.д.). Итого в результате ширина результирующего изображения будет равна сумме количества основных и дополнительных точек, иными словами соответствует равенству):

$$W + (W - 1)(N - 1) = N(W - 1) + 1$$

То есть, для последнего пикселя (в каждой строке и столбце) исходного изображения не находится пары, с которой можно было бы провести интерполирование. Так же этот метод иногда вызывает нежелательные эффекты сглаживания деталей, приводит к размытию изображения и всё равно порождает довольно заметный лестничный эффект. Однако, этот метод всё же масштабирует с приемлемым качеством даже при значительных изменениях масштаба, при этом вычислительная сложность низка, что положительно сказывается на скорости работы.

Развитием метода билинейной интерполяции является бикубическая интерполяция. Для вычисления цвета нового пикселя она использует квадрат размерностью не два на два пикселя, как билинейная, а четыре на четыре. Так как при таком подходе расстояния до каждого рассматриваемого пикселя являются различными, то вводится система весов, отражающих вклад цвета каждого из рассматриваемых пикселей в цвет искомого. Значения весов составляют $\frac{1}{4}$ для четырёх ближайших пикселей, $\frac{1}{36}$ для четырёх самых отдалённых и $\frac{1}{12}$ для всех остальных. Бикубическая интерполяция даёт более качественное изображение по сравнению с билинейной интерполяцией, однако вычислительная сложность этого метода намного выше. Подобным образом можно использовать и другие виды интерполяции, вычисляя значения функции по соседним $4k^2$ точкам, но качество этих формул хуже чем бикубической интерполяции.

Алгоритм направленной интерполяции основан на вычислении градиента цвета пикселя. Градиент — вектор, показывающий направление наискорейшего возрастания некоторой величины φ, значение которой меняется от одной точки пространства к другой (скалярного поля). Например, если взять в качестве φ значение цвета пикселя, то её градиент в каждой точке изображения будет показывать «направление границы цветов». Модуль вектора градиента равен скорости роста φ в этом направлении. Этот метод является упрощённой по части вычислений вариацией билинейной интерполяции, заменяющей сложную формулу вычислений значения функции цвета пикселя на вычисление градиента в точке назначения и соприкасающихся с ней точках изначального изображения, при условии увеличения изображения в 2 раза. Чтобы это осуществить, текущий обрабатываемый пиксель масштабируемого изображения условно делится на четыре новых.

В качестве примера алгоритма с предварительным анализом изображения алгоритм направленной интерполяции. Он основан на вычислении градиента цвета пикселя. Градиент — вектор, показывающий направление наискорейшего возрастания некоторой величины φ, значение которой меняется от одной точки пространства к другой (скалярного поля). Например, если взять в качестве φ значение цвета пикселя, то её градиент в каждой точке изображения будет показывать «направление границы цветов». Модуль вектора градиента равен скорости роста φ в этом направлении. Этот метод является упрощённой по части вычислений вариацией билинейной интерполяции, заменяющей сложную формулу вычислений значения функции цвета пикселя на вычисление градиента в точке назначения и соприкасающихся с ней точках изначального изображения, при условии увеличения изображения в 2 раза. Чтобы это осуществить, текущий обрабатываемый пиксель масштабируемого изображения условно делится на четыре новых. Наглядный пример отбора пикселей для вычисления градиента изображён на рисунке 2.

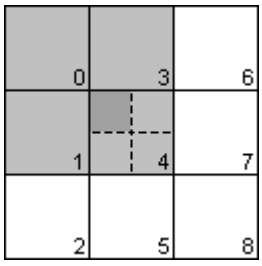

Рис.2. Пример отбора пикселей для вычисления градиента

Цветом на этом рисунке выделены пиксели, значения градиента цвета в которых будут использоваться для вычисления цвета одного из новых пикселей, получаемых из пикселя номер четыре, т.е. будут выбраны сам четвёртый пиксель и соприкасающиеся с ним пиксели номер один, ноль и три.

После вычисления модуля градиента в новом пикселе и соприкасающихся с ним, производится выбор минимального из значений градиента и итоговый пиксель получается путём интерполяции между новым пикселем и пикселем имеющим наименьшее значение градиента, в частном случае, когда наименьшее значение градиента находится в новом пикселе, он переносится на итоговое изображение без изменений. Так как масштаб изменялся в два раза то операция интерполяции, в свою очередь, сводится к вычислению среднего арифметического значений цвета двух пикселей.

Такой подход позволяет интерполировать вдоль границ, а не игнорировать их как в билинейной и бикубической интерполяции, что приводит к значительному увеличению чёткости изображения.

Этот метод не только отличается большим быстродействием чем билинейная интерполяция, но и позволяет значительно уменьшить «размытость» изображения. Недостатком же этого алгоритма является тот факт, что он позволяет работать только с увеличением изображения в два раза, т.е. для увеличения изображения в число раз равное степени двух придётся применить этот алгоритм к изображению несколько раз, и любое другое изменение масштаба изображения невозможно.

5. Оптимизация вычислений, проводимых алгоритмами масштабирования

Как видно, при масштабировании единовременно нет нужды во всём изображении, так как оно обрабатывается последовательно блоками в несколько пикселей. Точное число необходимых для вычислений пикселей, а так же сложность вычислений целиком зависит от конкретного применяемого алгоритма. Таким образом, все алгоритмы предполагают собой последовательную обработку изображения блок за блоком, что является не очень эффективным учитывая пределы разрешений изображения в современных системах отображения информации, особенно остро этот вопрос стоит в системах видеоотображения, которым необходимо обрабатывать и отображать информацию с высокой скоростью.

Часто, при использовании алгоритмов обрабатывающих изображения без предварительного анализа, для ускорения обработки используют заранее рассчитанные матрицы коэффициентов, указывающие в каких пропорциях смешивать цвета. Такой подход позволяет значительно ускорить обработку, так как пропускается этап вычисления коэффициентов масштабирования, однако он имеет два серьёзных недостатка:

1) для хранения предварительно рассчитанных матриц коэффициентов требуются большие объёмы памяти;

2) для каждой пары входного и выходного разрешения необходима отдельная матрица, то есть такой метод применим только для фиксированного набора разрешений.

Альтернативно, можно добиться значительно лучших результатов путём распараллеливания вычислительного процесса. Так как последовательно получаемы блоки пикселей обрабатываются полностью независимо друг от друга, то их распараллеливание не представляется сложным, при этом оно даёт линейное ускорение масштабирования, так обрабатывая по два блока за раз, процесс ускоряется ровно в два раза. Таким образом, если для масштабирования изображения неким линейным

алгоритмом потребуется время T, то для обработки его параллельной версией этого же алгоритма понадобится время T\n, где n – число параллельно обрабатываемых блоков, при этом должно выполняться неравенство n<N, где N – суммарное число блоков для обработки. Что примечательно, такой подход совместим с предыдущим, то есть можно добиться во много раз большей скорости обработки данных для фиксированного сочетания разрешений используя предрассчитанные матрицы и параллельную обработку. Таким образом можно будет добиться быстрой обработки даже с применением требовательных к вычислительным мощностям алгоритмов.

Но в данном направлении можно пойти дальше, применив каскадную обработку. В данном контексте это будет означать не единовременное изменение масштаба изображения в некоторое число раз, а постепенное масштабирование за несколько этапов. Из-за увеличения числа операций, прирост производительности может быть не очевидным, так как придётся снова и снова обрабатывать изображение. Но не стоит забывать о том, что для одной итерации цикла масштабирования, как уже упоминалось ранее, нет необходимости во всём изображении, только в его части, то есть можно будет организовать конвейер из нескольких параллельно работающих ступеней, каждая из которых обрабатывает часть изображения на определённом этапе. Например первая ступень конвейера увеличивает изображение в два раза и передаёт результаты работы на вторую, вторая же, в свою очередь, получив достаточное количество данных от первых ступеней увеличивает изображение ещё в два раза, как показано на рисунке 2.

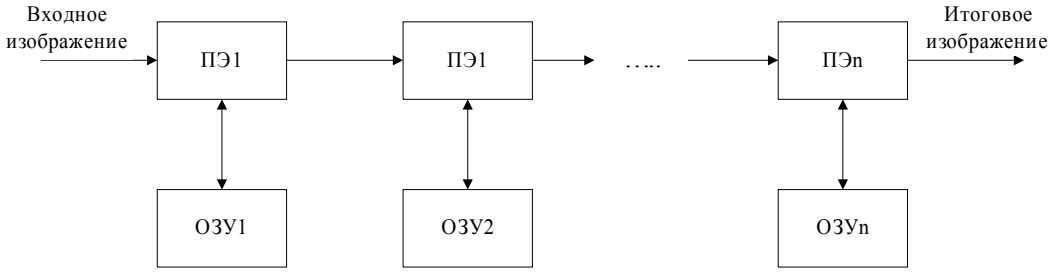

Рис. 2. Конвейерная обработка изображения

Изменение разрешения в два раза на каждом этапе обработки позволяет максимально упростить формулы, так как все обрабатываемы блоки при любом алгоритме масштабирования будут представлять собой квадраты, состоящие из равноправных между собой пикселей, что позволит использовать простые формулы расчётов, без ввода дополнительных коэффициентов, появляющихся при неровном наложении пиксельных сеток входного и выходного изображения. Как недостаток – выделение отдельного ПЭ под каждую ступень приводит к ограниченности допустимых пределов масштабирования. Можно попытаться избежать этого добавив обратную связь первого и последнего звеньев (как показано на рисунке 3), то есть позволить повторную обработку единожды масштабированного изображения, правда такой подход замедлит процесс масштабирования, что особенно критично при обработке изображений в реальном времени, например при работе с видео.

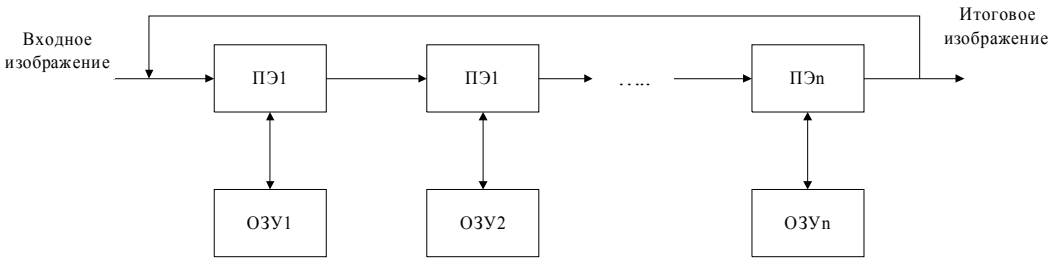

Рис. 3. Конвейерная обработка изображений

Эффективность такого подхода тем заметней, чем больше возможно параллельных ступеней обработки. Следовательно, для реализации подобного конвейера нужна элементная база с широкими возможностями по части распараллеливания вычислений, и достаточной производительностью для выполнения необходимых математических операций.

Конвейерные вычислители с динамически перестраиваемой архитектурой на базе FPGA (ПЛИС) являются перспективными элементами для построения мощных вычислительных систем, ускорительных секций суперкомпьютеров, ускорителей для персональных компьютеров, которые используются для решения сложных задач в науке, информационных технологиях, технике.

6. Общие сведения о ПЛИС

ПЛИС (программируемая логическая интегральная схема), – электронный компонент, предназначенный для создания цифровых интегральных схем. Логика работы ПЛИС задается посредством программирования, что позволяет реализовать на ней практически любую цифровую интегральную схему. Программирование осуществляется программатором, который заносит данные о структуре устройства в специальном откомпилированном формате, получаемом из описания электрической схемы или программы на языке описания аппаратуры, таком, как Verilog, VHDL или System C. Типовая структуры ПЛИС показан на рисунке 4.

Рис. 4. Типовая структура кристалла ПЛИС

Архитектурно ПЛИС состоит из матрицы конфигурируемых логических блоков (CLB – Configurable Logic Block), блоков ввода-вывода, модулей автоподстройки задержек (МАЗ), блочной памяти и соответствующих коммуникационных шин. Так же ПЛИС может включать в себя встроенные аппаратные блоки цифровых устройств (микропроцессоры, контроллеры интерфейсов, блоки математических операций и прочие).

Единичный конфигурируемый логический блок состоит из табличных преобразователей (LUT – Look-up Tables), реализующих запрограммированные функции, элементов памяти (триггеры) и коммутирующих мультиплексоров.

Таким образом базовая ячейка содержит как элементы памяти, так и элементы комбинационной логики. Конфигурируемые логические блоки объединяются в единую структуру при помощи конфигурируемых шин, что позволяет задать практически любую цифровую аппаратную схему. Пример конфигурируемого логического блока представлен на рисунке 5.

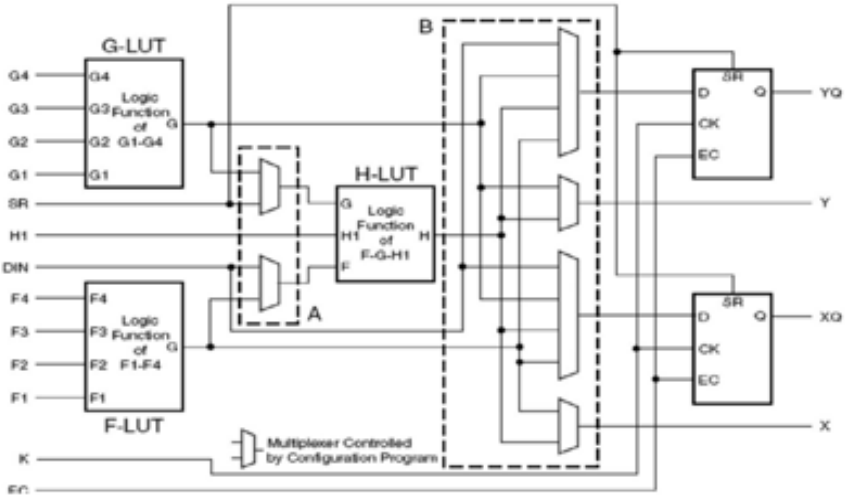

Рис 5. – Конфигурируемый логический блок (CLB).

Современные ПЛИС обладают значительными техническими характеристиками:

1) до миллиона конфигурируемых логических ячеек;
2) работа на тактовых частотах более 600 МГц;
3) наличие многовыводных корпусов (около 1000 выводов);
4) техпроцесс менее 32-нм позволяет достичь низкого энергопотребления и высокой степени интеграции;
5) наличие встроенных аппаратных ядер микропроцессоров (PowerPC, ARM), контроллеров памяти, шин (PCI-Express), периферийных устройств, сетевых интерфейсов (Ethernet).

В настоящий момент ПЛИС широко используются в различных задачах таких, как, цифровая обработка сигнала (ЦОС), цифровая видеоаудиоаппаратура, высокоскоростная передача данных, криптография, проектирование и прототипирование специализированных систем на кристалле, в качестве мостов (коммутаторов) между системами с различной логикой и напряжением питания, реализация нейрочипов.

7. Оценка эффективности

Основное преимущество по скорости работы конвейерного подхода, в сравнении с обработкой за один проход, не в малой мере является заслугой применения алгоритмов быстрого масштабирования с определённым коэффициентом (в нашем случае этот коэффициент равен двум), к таковым относятся большинство алгоритмов с предварительным анализом изображения. Такие алгоритмы, ускоряют обработку, за счёт полного отказа от операций умножения и деления, сводя всю к обработку к сложению, вычитанию и битовому сдвигу. Точное количество операций зависит напрямую от

выбранного алгоритма масштабирования, и, как правило, напрямую связано с качеством результата. В качестве примера рассмотрим алгоритм направленной интерполяции и билинейной интерполяции, как достаточно известные, с близким качеством и принципом работы, можно даже сказать что алгоритм направленной интерполяции является немного доработанным алгоритмом билинейной интерполяции.

Для вычисления цвета одного по алгоритму билинейной интерполяции нового пикселя требуется 12 операций умножения, четыре операции деления, 16 операций вычитания и три операции сложения, для получения результата подобные вычисления, для увеличения изображения разрешением X*Y в N раз, надо будет выполнить X*Y*N раз. При этом, чем сильнее коэффициент увеличения, тем более заметными будут артефакты масштабирования, так как используемые для расчётов пиксели изначального изображения будут всё более отдалёнными друг от друга на результирующем. Вдобавок к этому, придётся либо проводить существенно более медленные операции с числами с плавающей запятой, либо приносить в жертву качество работы, применяя целочисленные алгоритмы деления.

Вычисления по данному алгоритму можно распараллелить, так как алгоритм выполняет те же самые действия над четырьмя парами пикселей, а затем производит их сложение, то наиболее разумной представляется реализация параллельных вычислений для каждой пары, а затем их попарное сложение. Для подобной реализации понадобятся 4 ПЭ, при этом произойдёт заметное ускорение работы, так как вычисления для всех пар производятся одновременно, то затраченное время для получения одного нового пикселя будет равно времени для вычислений одной пары значений (вместо четырёх) и двух операций сложения. В итоге получается: 3 операции умножения, 4 операции вычитания и 2 операции сложения. Подобный подход применим и к другим подобным алгоритмам, основывающимся на единообразных для любого изображения операций над соседними пикселями.

При применении алгоритма направленной интерполяции совместно с конвейерной обработкой, для получения одного пикселя с выхода одной ступени требуется всего: три вычитания, одно сложение и деление на два. Для ускорения работы, деление результата на два может быть заменено на сдвиг вправо на один десятичный разряд, т.е. все вычисления сводятся к простым операциям сдвига и целочисленного сложения. Однако подобное возможно только при увеличении в два раза, для увеличения в N раз необходимо несколько ступеней, так как каждая из них работает с изображением разного размера, то и скорости работы ступеней будут разные. Так при увеличении изображения X*Y в N раз, в таком случае, понадобится обработать изображение Z раз на конвейере, где Z – целочисленный логарифм N по основанию 2. В итоге, на каждой ступени выполняется 4*X*Y*i вычислений нового пикселя, где i = (1,4,8,16,…), суммарное количество обработок будет равно Z. Для получения одного пикселя же надо выполнить на каждой ступени всего 3 операции вычитания, 3 операции сравнения и одну операцию сдвига, которые, следует отметить, являются целочисленными.

Таким образом достигается полный отказ от затратных операций умножения и деления и использование исключительно быстродействующих операций целочисленного сложения, вычитания и сдвига, которые, следует отметить, в современных ПЛИС не уступают по быстродействию даже БИС.

Однако, как можно заметить, количество данных для обработки растёт линейно в число раз равное квадрату коэффициента увеличения данной ступени (в нашем случае коэффициенты всех ступеней равны двум) раза на каждой на ступени, что приводит к неравномерной нагрузке конвейера. Возможным решением данной проблемы является дополнительное распараллеливание процесса обработки на каждой ступени, путём

введения дополнительных ПЭ, использующих совместно одно ОЗУ. Для поддержания равномерной нагрузки, число ПЭ должно увеличиваться на каждой ступени на квадрат коэффициента увеличения предыдущей ступени (т.е. в четыре раза). Такой подход может показаться нерациональным, но не стоит забывать, что сами по себе данные элементы довольно просты, следовательно увеличение их количества не должно быть проблематичным. Кроме того, при использовании ПЛИС возможно быстрое и простое добавление числа ПЭ в уже готовый проект.

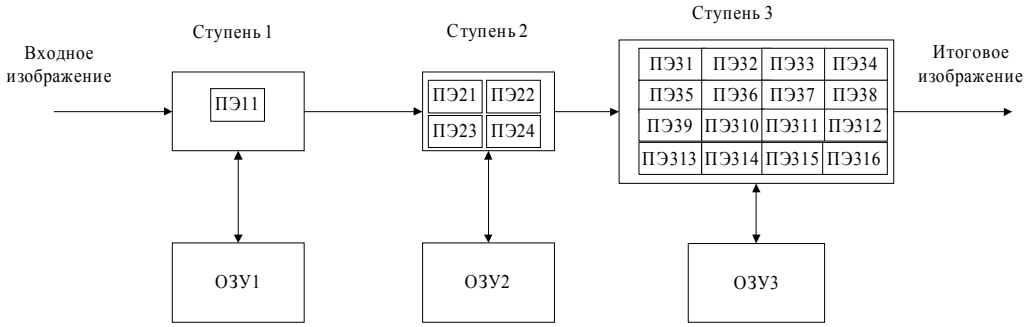

Рис 6. – Конвейер с параллельной обработкой на каждой ступени

Конечно же далеко не всегда масштаб изображения нужно изменить в число раз являющиеся степенью двух, следовательно последний каскад всё равно должен будет производить обработку с учётом неровного наложения пиксельных сеток. Однако, так как каждый из каскадов увеличивает изображение в два раза, то на последнем каскаде можно будет производить изменение разрешения на небольшую величину, меньшую двух. Для масштабирования в таких небольших пределах можно воспользоваться простым быстродействующим алгоритмом, не требующим сложных вычислений, например методом ближайшего соседа.

К сожалению, ввиду разного размера изображения на каждой итерации и, как следствие, разного времени обработки на каждой ступени (чем больший номер у ступени, тем больше времени занимает обработка), такой подход не слишком эффективен, когда известно уже всё изображение, и надо просто его обработать, т.е. в случае со статичным изображением. Видеопоток же получаемый в реальном времени, поступает не целиком, а частично, в таком случае, какая-либо задержка при обработке, компенсируется при условии, что изображение обрабатывается быстрее чем оно поступает. При современных скоростях передачи данных такое вполне достижимо на 1-3 ступенях обработки (восьмикратное увеличение) даже для достаточно высоких разрешений, в т.ч. для видео высокой чёткости. Такой подход позволяет сократить потери в скорости масштабирования при применении каскадов, позволяя в полной мере воспользоваться их преимуществами.

В итоге, применение параллельных вычислений при обработки изображения по алгоритмам, работающим независимо от обрабатываемого изображения, достигается большая скорость обработки при работе, однако большее изменение масштаба приводит к тому, что порождаемые интерполяцией артефакты становятся более заметны. С другой стороны алгоритмы, предварительно анализирующие масштабируемое изображение, как правило, работаю только с одним коэффициентом увеличения, так как сильно полагаются на цвета окружающих пикселей, и следовательно без использования конвейерной обработки крайне ограничены в области применения. В идеале, применяемый алгоритм должен зависеть от задачи. Применение же для масштабирования современных ПЛИС

позволит добиться как повышения быстродействия, в сравнении с чисто программным подходом, так и позволит сохранить возможность внесения изменений в систему.

8. Заключение

В статье рассмотрены:

1) Представление цифровых изображений в системах отображения информации
2) Примеры и анализ нескольких алгоритмов масштабирования.
3) Возможности оптимизации вычислений при масштабировании, путём применения параллельных вычислений и конвейерной обработки.
4) Современные ПЛИС.
5) Проведена оценка эффективности оптимизации масштабирования, путём применения конвейерной обработки.

В итоге, предлагается следующий подход к оптимизации процесса масштабирования, с применением параллельной обработки:

1) параллельная обработка нескольких блоков входного изображения;
2) постепенное каскадное масштабирование изображение, в два раза на каждой итерации;
3) параллельная конвейерная обработка всех ступеней каскада;
4) распараллеливание обработки на каждой ступени каскада;
5) применение простого быстродействующего алгоритма на последней ступени каскада.

В качестве аппаратной основы можно использовать ПЛИС.

СПИСОК ЛИТЕРАТУРЫ

1. *Егоров И.В., Внуков А.А.* Конвейерная обработка цифровых изображений при масштабировании // Тезисы V Международной научно-практической конференции «Инженерные системы — 2012». М.: РУДН. 2012. С. 41.
2. *Егоров И.В., Внуков А.А.* Быстрый алгоритм масштабирования изображений // Материалы международной научно-практической конференции Инновационные информационные технологии. М.: МИЭМ., 2012. С. 276-278.
3. *Schmidt M., Reichenbach1 M., Loos A., Fey D.Третий Т.Т.* A smart camera processing pipeline for image applications utilising marching pixels // Signal & Image Processing : An International Journal V.2, No.3, September 2011.