УДК 004.383.3 / 519.684.6 DOI: 10.22184/NanoRus.2019.12.89.73.77

АРХИТЕКТУРА NEUROMATRIX В ТРЕХМЕРНОЙ ГРАФИКЕ: ВОЗМОЖНОСТИ И ПЕРСПЕКТИВЫ

NEUROMATRIX ARCHITECTURE IN 3D-GRAPHICS: OPPORTUNITIES AND PROSPECTS

МУШКАЕВ СЕРГЕЙ ВИКТОРОВИЧ
mushkaev@module.ru

ЖИЛЕНКОВ ИВАН ВИКТОРОВИЧ

ЗАО НТЦ «Модуль»
125190, г. Москва, а/я 166

MUSHKAEV SERGEY V.
mushkaev@module.ru

ZHILENKOV IVAN V.

RC Module JSC
P.O. Box 166, Moscow, Russia, 125190

В докладе рассматривается архитектура NeuroMatrix с точки зрения применимости в компьютерной графике. Изучаются аппаратные возможности векторно-матричного целочисленного сопроцессора и сопроцессора с плавающей точкой в задачах геометрических преобразований 3D-объектов с последующим рендерингом. Предлагаются методы векторизации вычислений в определении проекций, растеризации, реализации Z-буфера и т. д. Анализируются эффективность и перспективы развития архитектуры NeuroMatrix в трехмерной графике.
Ключевые слова: поточная обработка данных; 3D-графика; параллельные вычисления; NeuroMatrix; рендеринг.
The paper considers architecture of NeuroMatrix in terms of applicability in computer graphics, as well as hardware capabilities of vector-matrix fixed-point and floating-point coprocessors in the problems of geometric transformations of 3D objects with subsequent rendering. Methods of vectorized computations in the definition of projections, rasterization, Z-buffer implementation, etc. have been proposed. The efficiency and prospects of the NeuroMatrix architecture development in 3D graphics have been analyzed.
Keywords: stream processing; parallel computation; BLAS; vector-matrix multiplication; NeuroMatrix; 3D graphics; rendering.

ВВЕДЕНИЕ

Реализация современных программ накладывает высокие требования к вычислительной аппаратуре. Одни вычисления больше подходят к универсальным процессорам, другие — к DSP. Как правило, большинство современных сложных программ содержат и те и другие. В таких ситуациях наилучшим решением становятся гетерогенные системы, содержащие как универсальные процессоры, так и DSP-ядра. Однако в ряде случаев подобные задачи ставятся и перед самостоятельными DSP-процессорами. В данном докладе рассматривается архитектура DSP NeuroMatrix, и для нее объемные скалярные вычисления, не поддающиеся прямой векторизации, могут стать серьезным бутылочным горлом. Тем не менее практика показывает, что зачастую даже такие программы все же поддаются частичной или даже полной векторизации с помощью алгоритмических методов. Несмотря на увеличенную сложность векторизованного алгоритма, общая производительность может оказаться выше более простого скалярного. Ниже будет рассмотрена задача рендеринга 3D-модели на процессоре NeuroMatrix. Данных класс задач не типичен для DSP, и в частности для NeuroMatrix. Главная проблема заключается в нерегулярности данных: разная форма и размер полигонов, алгоритмы заливки и растеризации с множеством условий и ветвлений, на первый взгляд, не позволяют задействовать векторно-матричное ядро. Тем не менее решение было найдено и заключается не в заливке треугольных областей классическими методами, а в формировании их из готовых шаблонов (паттернов). При таком методе используются простейшие логические и арифметические операции. Способность архитектуры целочисленного сопроцессора NeuroMatrix работать с упакованными данными произвольной разрядности вплоть до 1–2 бит позволяет не только векторизовать обработку, но и сделать ее эффективной и экономной по памяти. Формирование 1–2-битных масок, определяющих форму полигонов, в дальнейшем уже используется для растеризации нужным цветом и Z-буферизации. Остальные задачи по геометрическим преобразованиям (повороты, смещение, масштабирование и т. д.), вычисление проекций, а также вычисления освещенностей каждого полигона уже успешно решает сопроцессор с плавающей точкой.

ПОСТАНОВКА ЗАДАЧИ

Задачей данной работы было исследование применимости архитектуры NeuroMatrix в области трехмерной графики. Т.е. по имеющейся 3D-модели, составленной из треугольных полигонов, необходимо:

- осуществить движение модели в пространстве;
- определить освещенность полигонов в соответствии с положением источника света;
- найти прямолинейную проекцию 3D-координат на плоскость;
- по полученным значениям освещенности осуществить однородную растеризацию полигонов;
- определить видимости перекрывающихся полигонов.

Приведенные вычисления делятся на два класса: трансформация координат (в классической терминологии за нее отвечает вершинный шейдер), и растеризация — задача пиксельного шейдера. Работа с координатами осуществляется в арифметике с плавающей точкой, растеризация и определение

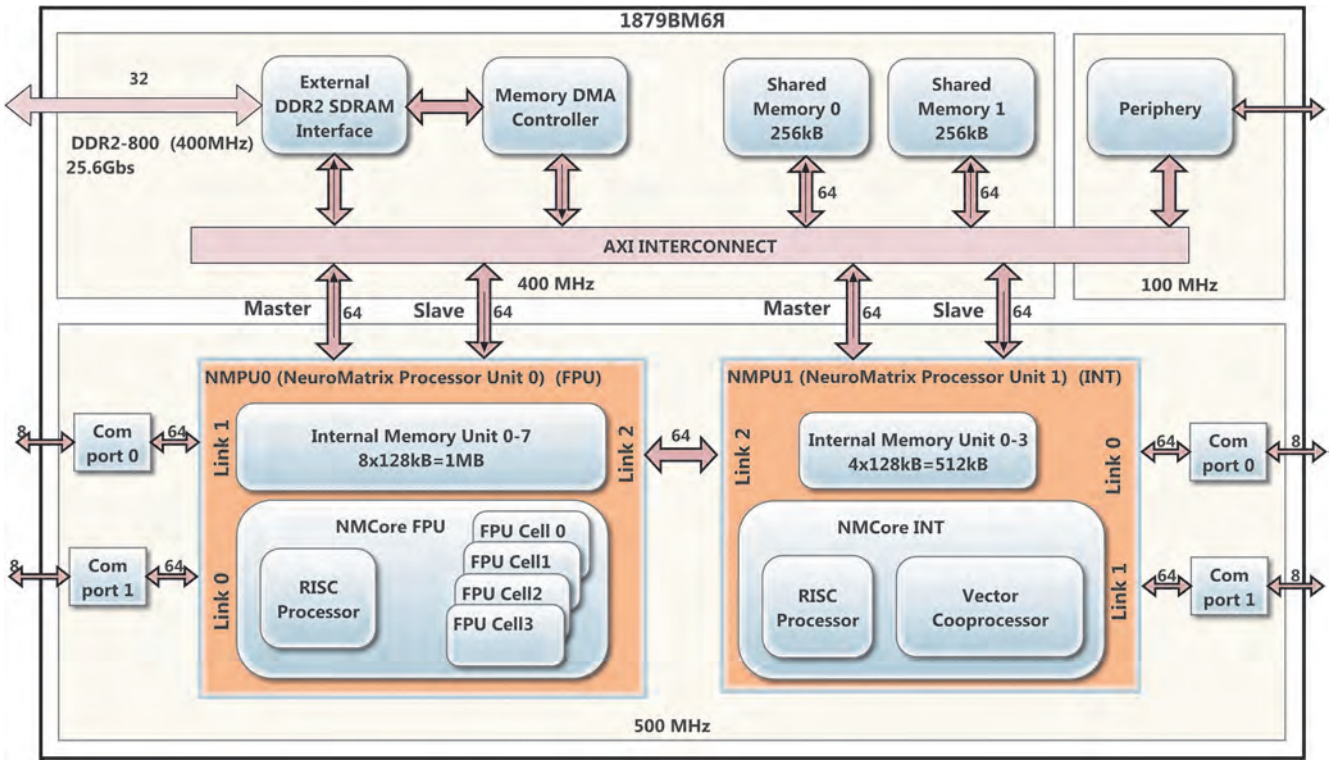

Рис. 1. Структура процессора 1879ВМ6Я

видимости — в целочисленной арифметике. Оба типа вычислений поддерживаются в процессоре 1879ВМ6Я [1]. Процессор содержит два независимых ядра для работы с плавающей и фиксированной точкой, см. рис. 1. Ядро с плавающей точкой содержит четыре вычислительных ячейки, каждая из которых способна выполнять матрично-векторное умножение 2×2 одинарной точности за один такт, либо скалярное умножение $a \cdot b + c$ двойной точности, либо комплексное одинарной точности, см. рис. 2. Выполняемые задачи распределяются согласно типу арифметики, на которой они основаны.

ЗАДАЧИ, РЕШАЕМЫЕ НА NPU0 (FPU)

На ядре с плавающей точкой (NMPU0) выполняются следующие действия:

- Движение модели осуществляется с помощью умножения на матрицу поворота 4×4.
- Проекция, в данном проекте аксонометрическая, выполняется на плоскость XY отбрасыванием Z-координаты.
- Сортировка вершин треугольников осуществляется при фрагментации сложных 3D-сцен на отдельные сегменты в целях размещения их в быстрой внутренней памяти.

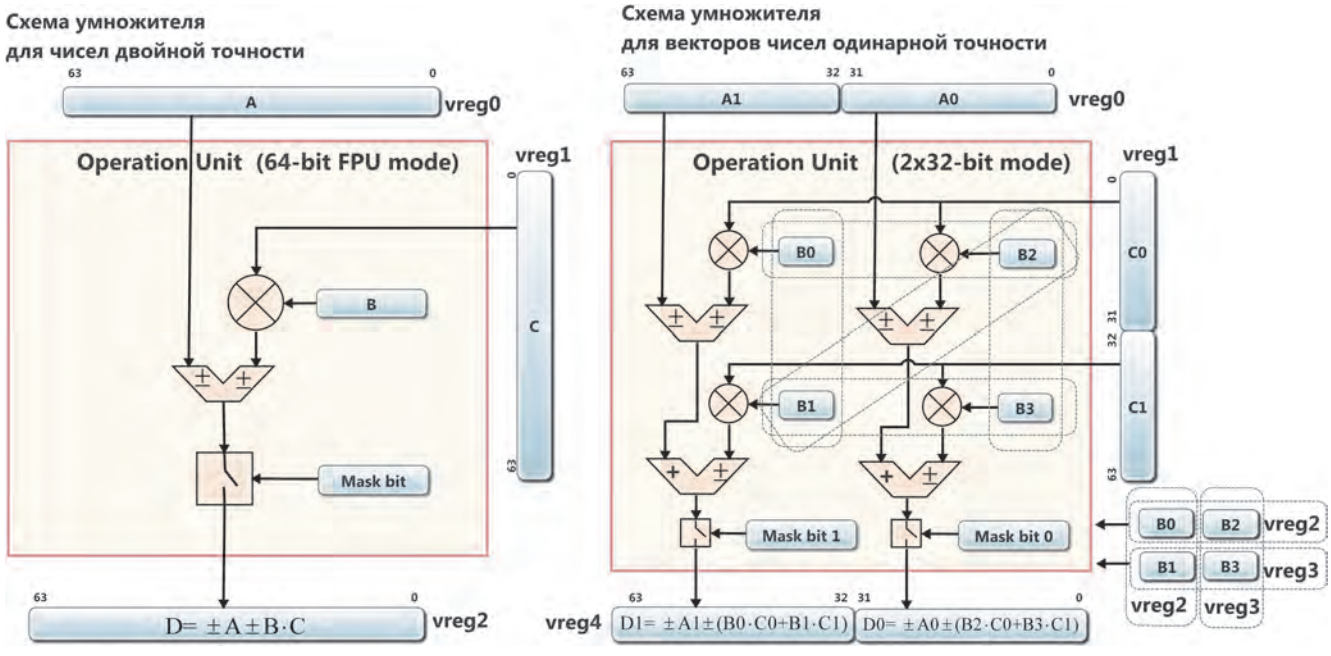

Рис. 2. Структура операционного узла

- Производится переупорядочение вершин (от верхней координаты по оси Y к нижней) внутри каждого треугольника, необходимого для алгоритма растеризации.
- С помощью векторного произведения сторон треугольника вычисляются нормали к плоскости полигонов. По направлению нормалей производится отсечение обратно-ориентированных к наблюдателю полигонов, не требующих отрисовки.
- С помощью скалярного произведения нормали с вектором освещения находится яркость каждого полигона.
- Полученные координаты и значения освещенностей преобразуются в целочисленной формат с помощью аппаратного блока упаковки и передаются на целочисленный процессор через разделяемую память. Перечисленные задачи основаны на стандартных математических операциях, хорошо векторизуются на ядре с плавающей точкой и реализованы в математической библиотеке NMPP [2] для данного процессора.

ЗАДАЧИ, РЕШАЕМЫЕ НА NPU1 (INT)

Оставшаяся обработка для целочисленного узла [2] состоит в растеризации рис. 3 и определении видимости каждого полигона. Задача наложения текстуры в данной работе не рассматривалась.

АЛГОРИТМ РАСТЕРИЗАЦИИ

Варианты растеризации классическими методом заливки треугольников содержат нерегулярные вычисления (используют уравнения прямых, циклы, операции ветвления) и, соответственно, возможны только на скалярном ядре. Ввиду невозможности задействовать векторный узел и нецелесообразности использования скалярного ядра такой подход был отброшен. В качестве альтернативного решения был разработан подход, заключающийся в предварительной генерации большого множества прямоугольных областей (~8000 паттернов), разделенных на две части наклонной прямой под всевозможными углами от 0 до 180 градусов и залитых только с одной стороны, см. рис. 4.

Далее для каждого ребра растеризуемого треугольника, зная его размер и наклон, производят извлечение нужного прямоугольного фрагмента из соответствующего паттерна. Окончательное формирование залитого треугольника ABC осуществляется путем логического объединения трех выбранных фрагментов от каждого ребра треугольника, см. рис. 5.

Для технической реализации алгоритма достаточно было бы иметь битовое представление паттернов и, соответственно, получить в результате логического объедения бинарную маску треугольника. Однако в дальнейшем эту маску необходимо растеризовать в соответствии с назначенной полигону яркостью L, применяя равномерную закраску, т. е. умножить каждый бит на яркость полигона L, а также умножить на значение глубины Z для определения видимости треугольника в алгоритме Z-буфера. Это приводит к необходимости умножать одноразрядные данные на константу. Большинство процессоров не поддерживает данную операцию, вынуждая хранить каждый бит маски минимум в 8-разрядной сетке, что явно неэффективно, так как приводит к 8-кратному увеличению объема памяти паттернов и к такому же падению производительности на логических операциях. Иначе ситуация обстоит с DSP-процессором с архитектурой NeuroMatrix. Отличительной особенностью данного процессора является его способность выполнять арифметические операции над упакованными данными произвольной разрядности. Векторный процессор оперирует 64-разрядными векторами, разбиение которых на отдельные элементы осуществляется программным способом. В арифметических операциях процессор оперирует только целыми данными со знаком, поэтому минимально возможным разбиением является двухбитовое. Таким образом, это приводит к необходимости хранить паттерны в двухбитовом формате, что делает описанный алгоритм достаточно эффективным и требует всего 2 MB памяти DDR на 8000 паттернов. Размер паттернов выбран равным 32×32 пикселям, это накладывает ограничения на максимальный размер полигонов: если исходный полигон выходит за границы, то его необходимо разбить на более мелкие треугольники.

Рассмотрим процесс умножения 2-битной маски полученного треугольника на константу L. Целочисленный сопроцессор имеет матрицу коэффициентов с программируемым разбиением и общим размером 64×64 бита. Процессор способен производить умножение с накоплением за один такт [2]. В данной задаче схема разбиения и умножения одной строки (элементы b0-b31) 2-битной маски треугольника на 16-разрядную константу показана на рис. 6. На каждом такте диагональная матрица wL_i из четырех коэффициентов L умножается на 64-р. вектор, содержащий строку маски b0-b31. В результате каждого умножения формируется вектор из четырех 16-разрядных произведений $b_i \cdot L$. Полное умножение маски 32×32 требует $32 \cdot 4 = 128$ тактов. Разрядность в 16 бит обусловлена тем, что ее достаточно как для реализации Z-буфера с максимальной глубиной 32768, так и для представления освещенности либо в grayscale-256, либо в формате RGB 5-5-6.

Аналогичным образом 2-битовая маска умножается на 16-разрядное значение глубины Z. Таким образом, с помощью описанной выше схемы для каждого полигона находятся два 16-разрядных прямоугольных фрагмента, содержащих

Рис. 3. Растеризация

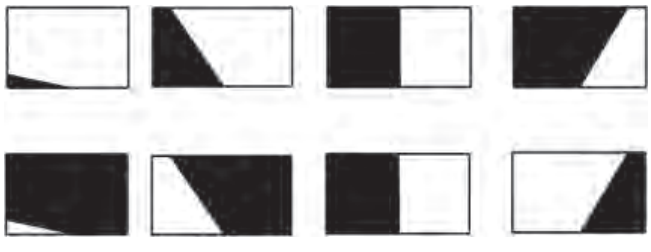

Рис. 4. Набор паттернов

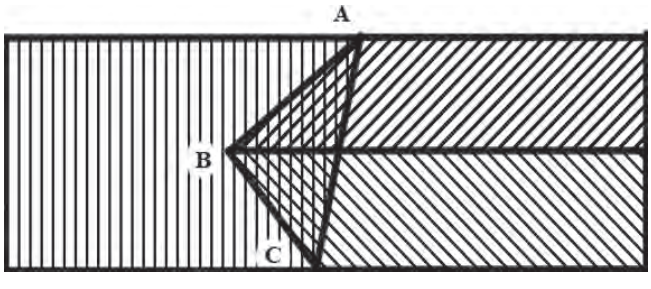

Рис. 5. Формирование залитого треугольника

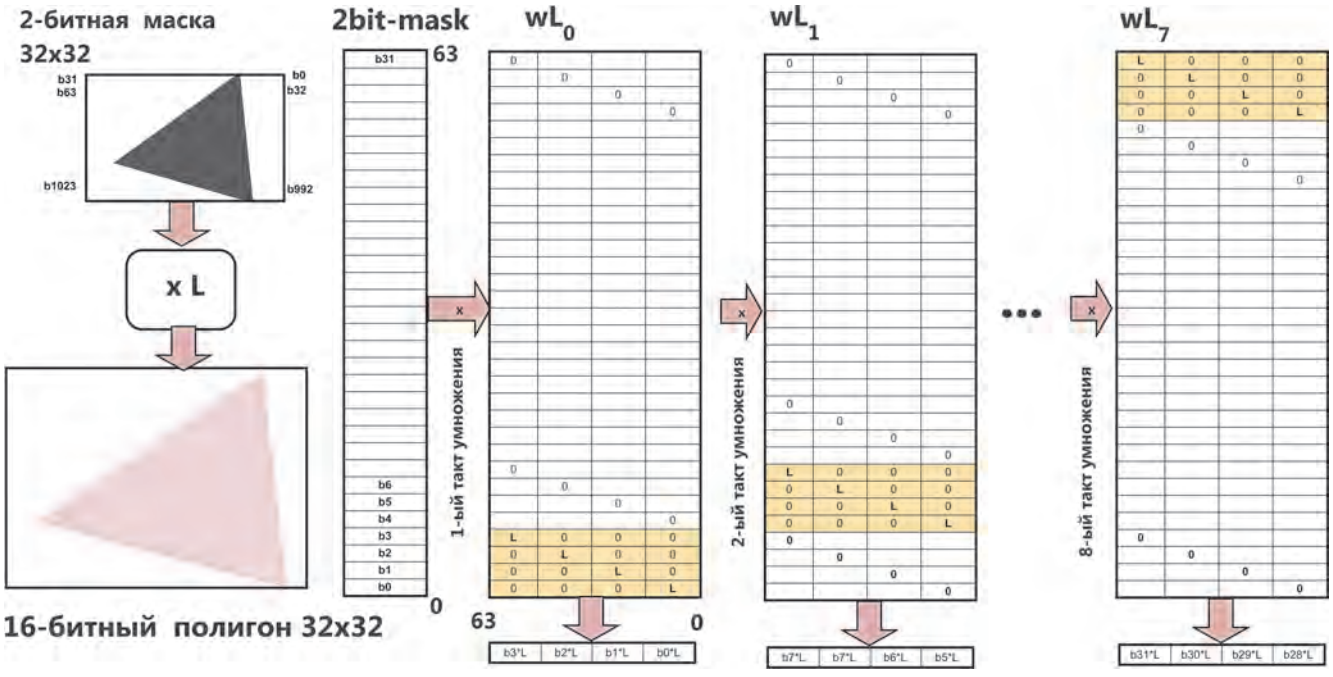

Рис. 6. Умножение 2-битной маски на константу

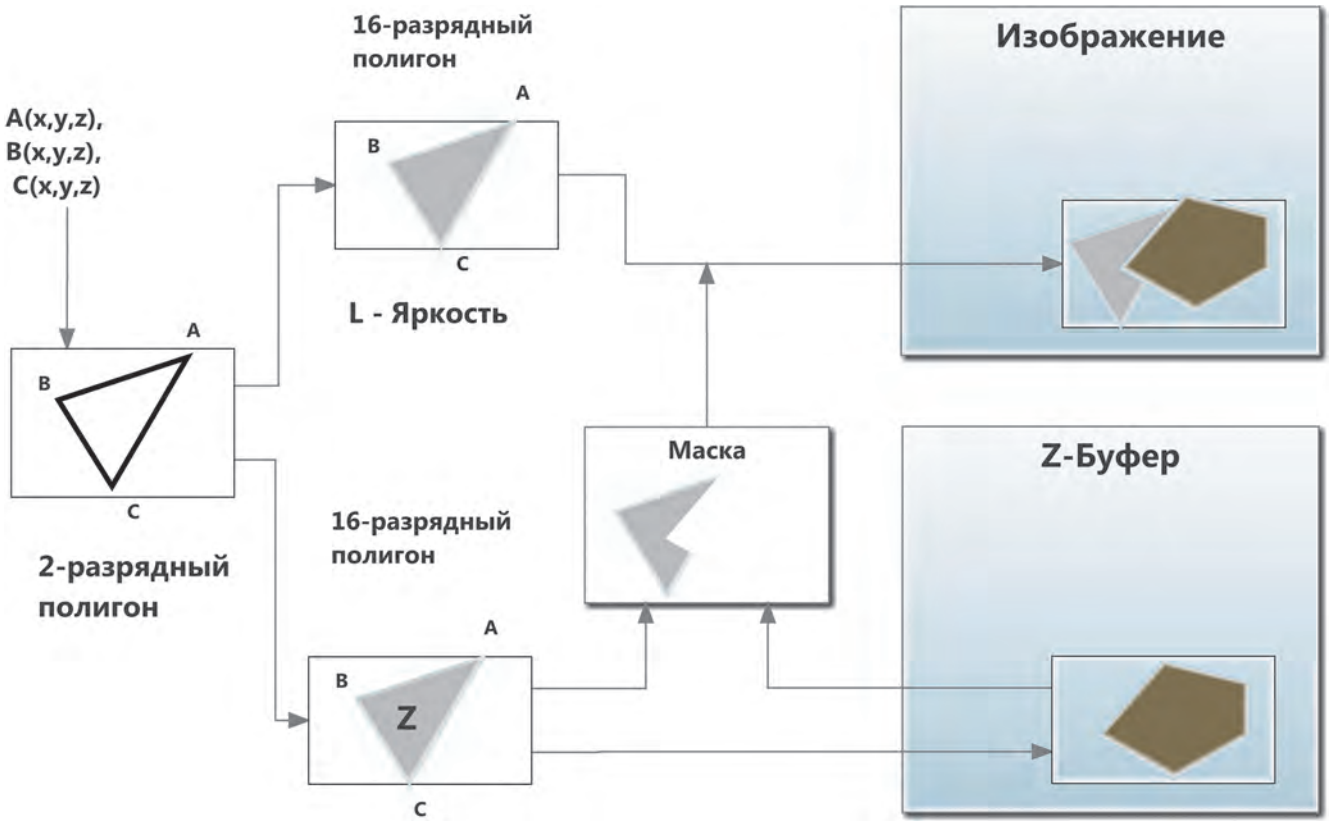

Рис. 7. Растеризация и Z-буферизация

треугольники, заполненные значениями L и Z. Далее путем сравнения полученного Z-фрагмента с состоянием Z-буфера находится маска видимости, по этой маске на результирующее изображение накладывается L-фрагмент, см. рис. 7, а состояние Z-буфера обновляется. Векторный процессор аппаратно поддерживает однотактовую операцию маскирования Z = (X and M) or (Y and not M), что также дополнительно ускоряет алгоритм.

Описанный выше алгоритм делает принципиально возможной векторизацию рендеринга и его реализацию на NeuroMatrix. Накристальная память целочисленного векторного ядра составляет 512 Кб, позволяет хранить Z-буфер 256×256 и обрабатывать до 2000 полигонов, накапливая их в сегменте изображения 256×256. Чистое среднее время растеризации одного видимого (обращенного к наблюдателю) полигона с решением задачи его перекрытия другими полигонами

Таблица 1. Производительность рендеринга на процессоре 1879ВМ6Я (500 МГц)

Кол-во отрисованных (видимых) полигонов из общего числа	Тактов	FPS	Тактов на полигон	Тактов на точку при средней площади полигона 80 пикселей	Полигонов в сек. (grayscale)	Полигонов в сек. (RGB 24)
448 из 896 (256×256)	150000	350	3200	40	160 000	50 000
20000 (720×576)	96000000	5	4800	60	100 000	33 000

составляет 1700 таков, что при средней площади полигона 80 пикселей составляет примерно 20 тактов на пиксель.

Для обработки сцены с большим числом полигонов необходимо произвести предварительную фрагментацию сцены на сегменты 256×256. Как было упомянуто выше, данную операцию сортировки полигонов по их координатам достаточно быстро производят на векторном процессоре с плавающей точкой. Операция на нулевом процессоре NMPU0 выполняется на фоне работы целочисленного ядра NMPU1 и не влияет на суммарное время. Однако, чтобы полигоны не попадали на границу сегментов, фрагментацию необходимо производить с перекрытием. Также необходимо обеспечить синхронизацию и обмен данными между ядрами, выгрузку во внешнюю DDR-память. Несмотря на привлечение DMA для копирования данных, указанные накладные расходы оказывают влияние и среднее врем обработки полигона возрастает до 4800 тактов на полигон или 60 тактов на пиксель. В табл. 1 приведены полученные данные производительности по полной задаче движения и рендеринга 3D-модели на одном процессоре 1879ВМ6Я.

ЗАКЛЮЧЕНИЕ

Предоставляемый пакет средств разработки позволил успешно реализовать предложенный алгоритм на отладочном модуле МЦ121.01 с процессором 1879ВМ6Я. В проекте были задействованы механизмы распараллеливания вычислений, пакетные режимы ПДП, управление многобанковой структурой памяти, механизмы межпроцессорного обмена, в том числе с использованием циклических буферов, применялась фрагментация для работы с большим объемом данных через внутреннюю память. Межпроцессорный обмен, синхронизация и взаимодействие модуля с host-компьютером осуществлялись по каналу USB с помощью библиотеки HAL и библиотеки загрузки и обмена. Управление и отображение графических данных осуществлялось с помощью библиотеки VSHELL, а вся алгоритмическая часть программы построена на функциях математической библиотеки NMPP [3]. Дополнительного ускорения удалось добиться оптимизацией некоторых функций на ассемблере под векторный процессор. Реализация данного проекта, а также опыт других задач демонстрируют, что в ряде случаев нерегулярные вычисления поддаются частичной либо полной векторизации алгоритмическими методами, что резко повышает эффективность вычислений и позволяет решить задачи чисто программными средствами. Во многих случаях такая возможность обеспечивается за счет гибкой программируемой структуры векторного сопроцессора, позволяющего производить как битовые манипуляции, так и арифметические действия над векторными данными произвольной разрядности то 1 до 64 бит. Несмотря на нетипичность задачи рендеринга для DSP-процессоров, была показана применимость процессора NeuroMatrix в таком классе задач. Полученные оценки производительности (40–60 тактов на пиксель на одном процессоре в формате grayscale или RGB (5-6-5)) показывают, что архитектура процессора обладает определенным потенциалом в области трехмерной графики и, в частности, возможна поддержка OpenGL. Следует отметить, что достигнутая производительность не является окончательной, так как остается достаточно большое поле для дальнейшей оптимизации. При соответствующем дальнейшем развитии архитектуры и поддержке специальными аппаратными решениями, в том числе многоядерными, процессор может стать высокоэффективным и производительным вычислителем в области компьютерной графики.

ЛИТЕРАТУРА

1. Черников А. В., Черников В. М., Виксне П. Е., Шелухин А. М. Высокопроизводительное процессорное ядро NMC4 для обработки векторных данных в форматах с плавающей и фиксированной точками // Сборник докладов 3-й Международной научной конференции «Электронная компонентная база и электронные модули». URL: https://www.module.ru/upload/images/1543575869NMC4HighPerfVectorProcessingCore.pdf.
2. Черников В. М., Виксне П. Е., Шелухин А. М., Шевченко П. А., Панфилов А. П., Косоруков Д. Е., Черников А. В. Семейство процессоров обработки сигналов с векторно-матричной архитектурой NeuroMatrix // Электронные компоненты, 2006. — № 6. — С. 79–84.
3. URL: https://github.com/RC-MODULE/nmpp.