УДК 004.2

АРХИТЕКТУРА СБИС ГРАФИЧЕСКОГО УСКОРИТЕЛЯ С ВОЗМОЖНОСТЬЮ ПОДДЕРЖКИ СТАНДАРТОВ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Тарасов И.Е.

МИРЭА - Российский технологический университет, 119454, Россия, г. Москва, проспект Вернадского, 78, e-mail: tarasov_i@mirea.ru

В статье анализируются современные подходы и стандарты описания вычислений общего назначения для графических ускорителей согласно подходу GPGPU (General Purpose computing on Graphics Processing Unit). Рассматриваются перспективы реализации сверхбольшой интегральной схемы, ориентированной на параллельные вычисления в соответствии с задачами общего назначения, поддерживающей при этом построение отображение графических изображений в качестве вспомогательной функции. Аналогично GPU Nvidia Quadro, такие СБИС могут применяться для рабочих станций, выполняющих вычислительные задачи вида САПР, решения систем дифференциальных уравнений, спектрального и вейвлет-анализа, статистической обработки данных, машинного обучения и подобных классов вычислений, выполняющихся на большом количестве параллельных вычислительных узлов. Наличие стандартных подходов к описанию вычислений общего назначения позволяет основать процесс проектирования на существующих высокоуровневых моделях вычислений, сфокусировавшись на их аппаратной поддержке. При этом производительность в задачах, характерных только для построения трехмерных изображений, может быть уменьшена за счет повышения эффективности СБИС в вычислениях общего назначения.

Ключевые слова: процессор, графический ускоритель, СБИС, параллельные вычисления, аппаратная архитектура.

ARCHITECTURE OF A GRAPHICS ACCELERATOR VLSI WITH PARALLEL COMPUTING STANDARDS SUPPORT

Tarasov I. E.

MIREA - Russian Technological University, 119454, Moscow, 78 Vernadskogo Avenue, Russia, e-mail: tarasov_i@mirea.ru

The article analyzes modern approaches and standards for describing general-purpose computing for graphics accelerators according to the GPGPU (General Purpose computing on Graphics Processing Unit) approach. The prospects for the implementation of an ultra-large integrated circuit oriented towards parallel computing in accordance with general-purpose tasks, while supporting the construction of the display of graphic images as an auxiliary function, are considered. Similar to the Nvidia Quadro GPU, such VLSI can be used for workstations that perform computational tasks such as CAD, solving systems of differential equations, spectral and wavelet analysis, statistical data processing, machine learning and similar classes of calculations running on a large number of parallel computing nodes. The presence of standard approaches to the description of general-purpose computing makes it possible to base the design process on existing high-level computing models, focusing on their hardware support. At the same time, performance in tasks that are typical only for the construction of three-dimensional images can be reduced by increasing the efficiency of VLSI in general-purpose computing.

Keywords: processor, GPU, VLSI, parallel computing, hardware architecture.

Введение

Возникающие в настоящее время вопросы обеспечения технологического суверенитета требуют поиска перспективных направлений замены импортной элементной компонентной базы (ЭКБ) вычислительных систем. При широчайшей потребности в ЭКБ остаются актуальными вопросы технологического отставания и наличия большого количества проектов и в конечном итоге потенциальных потребителей создаваемых изделий. Вариантом широко востребованной элементной базы являются графические ускорители (Graphics Processing Unit, GPU), которые в настоящее время применяются также для реализации вычислений общего назначения определенных классов (GPGPU, General Purpose computing on Graphics Processing Unit). Этому способствует специфика требований к вычислительной платформе для построения трехмерных изображений – большое

количество специализированных вычислительных ядер и распределенная память. Распространенными технологиями вычислений на базе GPU в настоящее время являются Nvidia CUDA [1, 2, 3], OpenACC [4], OpenCL [5]. В рассмотренных стандартах применяется разделение памяти в соответствии с тем, как это реализовано в GPU. В современных реализациях используется иерархия памяти, от приватной (доступной отдельному вычислительному ядру) и локальной (доступной группе ядер) до глобальной (доступной всем ядрам). Ввиду этого подобную модель памяти следует сохранять в архитектурном понимании системы во избежание потери совместимости с распространенными моделями параллельных вычислений.

Архитектура СБИС графического ускорителя

Рассмотрение архитектуры СБИС графического ускорителя производится как с учетом возможностей поддержки распространенных стандартов, так и с учетом доступных российским дизайн-центрам технологических процессов и ограниченного потенциального тиража изделий по сравнению с мировыми лидерами в области GPU. В частности, исходными условиями следует принимать ограниченную площадь кристалла СБИС (т.е. существенно меньшее количество логических элементов), пониженную частоту относительно наиболее передовых технологических норм и затруднения с интеграцией высокоскоростных внешних интерфейсов и сложнофункциональных блоков. Поэтому при проектировании архитектуры необходимо в качестве приоритета выбрать поддержку актуальных вычислительных задач в области измерительной техники, цифровой связи, обработки данных, машинного обучения, цифрового моделирования и подобных классов вычислений, которые в настоящее время реализуются в том числе с привлечением GPU в качестве ускорителя.

При этом поддержка трехмерной графики может быть ограничена актуальными задачами промышленности, в частности, САПР, а визуальные эффекты, характерные для трехмерной игровой графики, могут быть не поддержаны или поддержаны по остаточному принципу. Такой подход позволяет частично скомпенсировать технологическое отставание, исключая из СБИС возможности, актуальные прежде всего для игровых приложений.

При реализации различных вариантов преобразований данных возрастает объем работ по функциональной верификации вычислительных устройств. Возникают риски внесения ошибок на интеграционном уровне, когда формально корректные субмодули окажутся некорректно взаимодействующими с точки зрения реализуемых алгоритмов, включая не столько ошибки в соединении сигналов, сколько неучтенные латентности, зависимости по данным, сложные условия с взаимным влиянием состояний различных субмодулей и т.д. В подобном случае наличие программируемого компонента позволяет в большой степени управлять процессом преобразований данных за счет перезагрузки программного обеспечения вычислительных элементов. Такой подход рассматривался, например, для графического ускорителя Intel Larrabee [6], однако не был реализован для серийно выпускаемых графических ускорителей. В настоящее время аналогичные проекты ведутся для процессорной архитектуры RISC-V [7, 8]. Представляется целесообразной разработка специализированного процессора, для которого, начиная с этапа архитектурного проектирования, учитывалась бы специфика рассматриваемого проекта (в то время как для проектов на базе RISC-V используются расширения команд процессорного ядра). В частности, специализация процессорного ядра за счет сокращения ресурсов общего назначения позволит разместить большее количество ядер на единицы площади СБИС.

Полнофункциональная поддержка стандартов параллельных вычислений общего назначения достаточно сложна и создает в процессе разработки высокие риски внесения логических ошибок в схему, которые не позволят использовать СБИС в определенных задачах. Для снижения таких рисков аппаратные реализации алгоритмов следует заменять программными, оставляя за аппаратными ускорителями базовые вычисления, имеющие очевидное назначение и порядок использования. Другой проблемой является распределение задач между вычислительными ресурсами, где некорректная реализация может также исключить возможность использования СБИС для определенных применений. Поэтому предлагается программная реализация процесса управления работой СБИС, для чего в ее состав необходимо ввести программируемый управляющий процессор, имеющий доступ к памяти программ процессорных элементов, выполняющий основной объем вычислений. Таким образом, в процессе работы СБИС появится возможность загружать как управляющее программное обеспечение, так и программы для основного массива процессорных элементов, причем выполняя это как передавая требуемые программы через управляющее ядро, так и запуская на управляющем ядре соответствующие программы, управляющие обновлением памяти программ процессорных элементов.

Архитектура вычислительного кластера СБИС графического ускорителя с поддержкой параллельных вычислений общего назначения показана на рис. 1. Гранулярная структура современных GPU накладывает определенные ограничения на исполняемые алгоритмы. В частности, иерархическая память подразумевает, что произвольное распределение памяти, принадлежащей отдельным ядрам (private memory), оказывается технически невозможным.

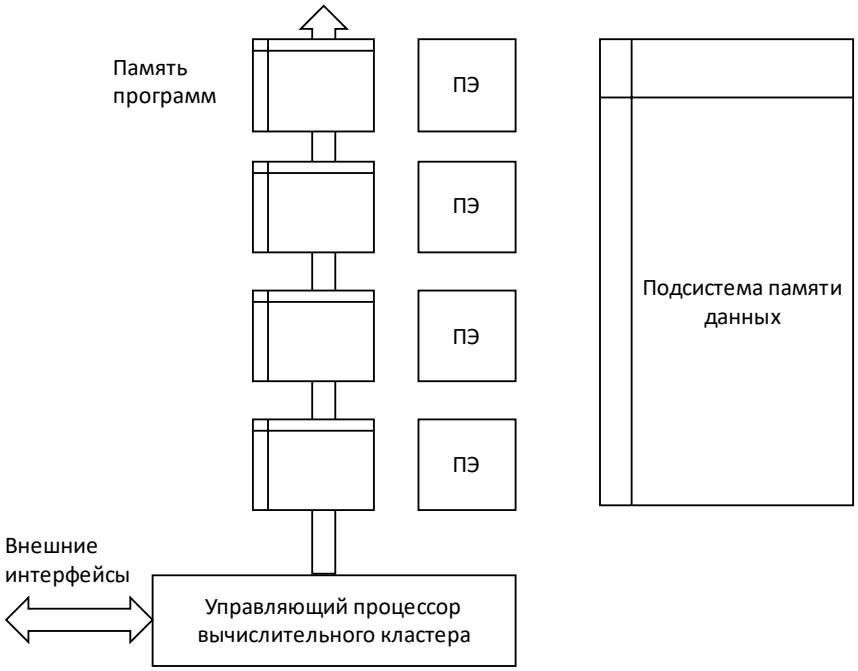

Рис.1. Архитектура вычислительного кластера СБИС графического ускорителя с поддержкой параллельных вычислений общего назначения

Наличие полного коммутатора для блоков памяти программ позволяет, в частности, реализовать подход Single Instruction, Multiple Data (SIMD), подключив несколько процессорных элементов к одному и тому же блоку памяти. В то же время, при наличии сложных алгоритмов, реализуемых с привлечением больших объемов памяти, появляется возможность их исполнения в рамках вычислительной структуры, с понижением общей производительности из-за неактивности процессорных элементов, которым не хватило памяти программ.

Таким образом, отдельные кластеры вычислительных узлов могут быть построены с применением коммутаторов, расширяющих возможности перераспределения локальной памяти между вычислительными ядрами, вплоть до реализации полных коммутаторов. Пример такой системы показан на рис. 2.

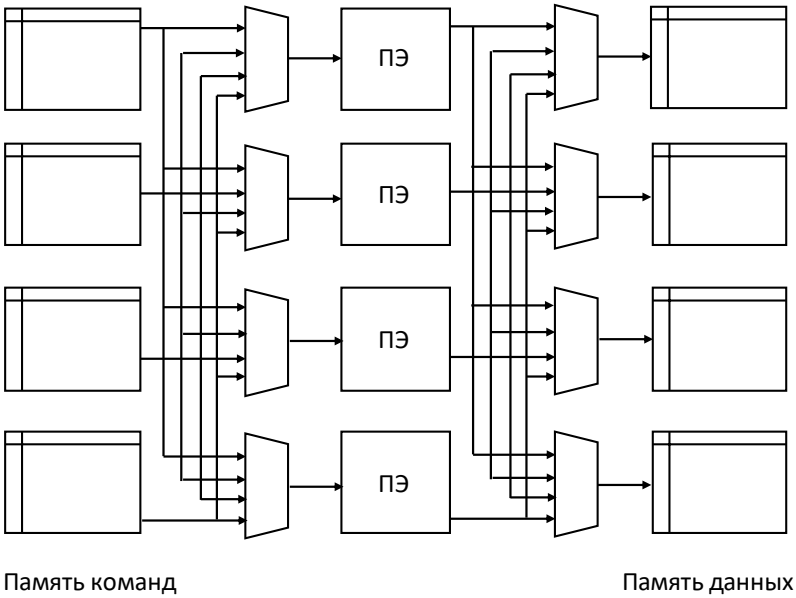

Рис.2. Подключение памяти в вычислительном кластере СБИС графического ускорителя

В показанной схеме процессорные элементы (ПЭ) имеют возможность доступа к любому фрагменту памяти данных, которая используется для хранения построенного изображения или в качестве памяти данных, используемых для ускорения вычислений общего назначения. Аналогично, любой из модулей памяти программ может быть подключен к любому из ПЭ. Таким образом, при нехватке памяти для выполнения определенного

алгоритма, память кластера может быть объединена и передана меньшему количеству ПЭ, вплоть до одного, с соответствующим уменьшением производительности вычислений, однако сохраняя при этом принципиальную возможность выполнения заданного алгоритма. Подобный подход не эквивалентен размещению в кластере памяти, локальной для этого кластера, поскольку при фрагментированной организации увеличивается общая доступная пропускная способность памяти.

Заключение

Предложенная архитектура открывает возможность проектирования серии СБИС, ориентированных преимущественно на вычисления общего назначения, для которых поддержка графической подсистемы является второстепенной задачей с точки зрения системной архитектуры и возможностей. Однако поддержка двумерной и частично трехмерной графики открывает перспективы широкого применения таких СБИС в устройствах различного назначения, от встраиваемых систем до высокопроизводительных рабочих станций. В качестве отличительных черт предлагаемой архитектуры можно указать полностью программируемые вычислительные узлы, включая узел распределения задач и фрагментированную память локальных кластеров с коммутируемым доступом, что обеспечивает возможность гибкого перераспределения ресурсов памяти внутри кластера для адаптации СБИС к задачам, отличным от построения графики.

Список литературы

1. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. – М.: ДМК Пресс, 2019. – 232 с.: ил. ISBN 978-5-97060-715-2
2. Тоуманен Б. Программирование GPU при помощи Python и CUDA / пер. с анг. А.В. Борескова. – М.: ДМК Пресс, 2020. – 264 с.: ил.
3. Сандерс Дж., Кэндрот Э. Технология CUDA в примерах: введение в программирование графических процессоров: Пер. с англ. Слинкина А.А., научный редактор Боресков А.В. – М.: ДМК Пресс, 2018. – 232 с.: ил. ISBN 978-5-97060-581-3
4. What is OpenACC? https://www.openacc.org/ (дата обращения 30.12.2022).
5. Антонюк В. А. OpenCL. Открытый язык для параллельных программ. – М.: Физический факультет МГУ им. М.В. Ломоносова, 2017. – 88 с.
6. Larry Seiler, Doug Carmean, Eric Sprangle и др. Larrabee: A Many-Core x86 Architecture for Visual Computing. ACM Transactions on Graphics, Vol. 27, No. 3, Article 18, Publication date: August 2008.
7. Elsabbagh, Fares & Tine, Blaise & Roshan, Priyadarshini & Lyons, Ethan & Kim, Euna & Shim, Da Eun & Zhu, Lingjun & Lim, Sung & kim, Hyesoon. (2020). Vortex: OpenCL Compatible RISC-V GPGPU.
8. Tine, Blaise & Elsabbagh, Fares & Yalamarthy, Krishna & Kim, Hyesoon. (2021). Vortex: Extending the RISC-V ISA for GPGPU and 3D-GraphicsResearch.

References

1. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. – М.: ДМК Пресс, 2019. – 232 с.: ил. ISBN 978-5-97060-715-2
2. Тоуманен Б. Программирование GPU при помощи Python и CUDA / пер. с анг. А.В. Борескова. – М.: ДМК Пресс, 2020. – 264 с.: ил.
3. Сандерс Дж., Кэндрот Э. Технология CUDA в примерах: введение в программирование графических процессоров: Пер. с англ. Слинкина А.А., научный редактор Боресков А.В. – М.: ДМК Пресс, 2018. – 232 с.: ил. ISBN 978-5-97060-581-3
4. What is OpenACC? https://www.openacc.org/ (дата обращения 30.12.2022).
5. Антонюк В. А. OpenCL. Открытый язык для параллельных программ. – М.: Физический факультет МГУ им. М.В. Ломоносова, 2017. – 88 с.
6. Larry Seiler, Doug Carmean, Eric Sprangle и др. Larrabee: A Many-Core x86 Architecture for Visual Computing. ACM Transactions on Graphics, Vol. 27, No. 3, Article 18, Publication date: August 2008.
7. Elsabbagh, Fares & Tine, Blaise & Roshan, Priyadarshini & Lyons, Ethan & Kim, Euna & Shim, Da Eun & Zhu, Lingjun & Lim, Sung & kim, Hyesoon. (2020). Vortex: OpenCL Compatible RISC-V GPGPU.
8. Tine, Blaise & Elsabbagh, Fares & Yalamarthy, Krishna & Kim, Hyesoon. (2021). Vortex: Extending the RISC-V ISA for GPGPU and 3D-GraphicsResearch.