ПОРТФЕЛЬ РЕДАКЦИИ

БИТ

N. P. Vasilyev, M. M. Rovnyagin, A. A. Skitev

The Methods of Implementation of the Three-dimensional Pseudorandom Number Generator DOZEN for Heterogeneous CPU/GPU/FPGA High-performance Systems

Key words: pseudorandom number generator, architecture "Square", architecture "Cube", parallel computing, hybrid architecture, NVIDIA CUDA, FPGA

The paper describes the scope of information security protocols based on PRNG in industrial systems. A method for implementing three-dimensional pseudorandom number generator DOZEN in hybrid systems is provided. The description and results of studies parallel CUDA-version of the algorithm for use in hybrid data centers and high-performance FPGA-version for use in hardware solutions in controlled facilities of SCADA-systems are given.

Н. П. Васильев, М. М. Ровнягин, А. А. Скитев

СПОСОБЫ РЕАЛИЗАЦИИ ТРЕХМЕРНОГО ГЕНЕРАТОРА ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ DOZEN В ГИБРИДНЫХ CPU/GPU/FPGA ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ СИСТЕМАХ

Введение

В настоящее время генераторы псевдослучайных чисел (ГПСЧ) используются при решении широкого спектра задач. В зависимости от характеристик ГПСЧ могут применяться: в вычислительной математике для решения задач методом Монте-Карло, в задачах имитационного моделирования систем массового обслуживания, в криптографии, в задачах верификации и др. При этом к генераторам, используемым для построения алгоритмов защиты информации, предъявляются наиболее жесткие требования. Такие ГПСЧ должны быть непредсказуемыми, статистически безопасными, а также формировать последовательности с большим периодом. Одновременно с этим криптографические алгоритмы предъявляют высокие требования к вычислительным ресурсам компьютеров.

Существует масса специализированных промышленных решений, ориентированных на использование в центрах обработки данных (ЦОД), автоматизированных системах управления технологическими процессами (АСУ ТП), системах с повышенными требованиями по защите конфиденциальной информации и персональных данных. В промышленных системах, например SCADA [1], на контролируемых объектах используется надежное оборудование, представляющее собой аппаратные решения, устойчивые к воздействию внешних факторов, имеющие компактные размеры и стандартные виды креплений (например, на DIN-рейку). В ЦОД, осуществляющих

обработку поступающих с объектов данных и расчет управляющих воздействий, применяются стандартные вычислительные решения с менее серьезными требованиями к условиям эксплуатации. При этом необходимо не только шифровать передаваемые по сети данные, но и обеспечивать защиту информации, выгружаемой во внешнюю память для долговременного хранения. Для этих целей также используются специализированные аппаратно-программные решения, начиная с поддержки шифрования и разграничения прав на уровне гипервизора подсистемы виртуализации [2] и заканчивая использованием сертифицированных в соответствии с FIPS жестких дисков [3]. До последнего времени кластеры с CPU-архитектурой справлялись с обработкой поступающих данных, а там, где это необходимо, использовались аппаратные решения. Но в настоящее время, в связи со значительным увеличением объемов обрабатываемых данных, все большее число ЦОД переходит с классической на гибридную CPU/GPU-архитектуру организации вычислений.

Архитектура современных высокопроизводительных систем

За последнее время архитектура высокопроизводительных систем хранения и обработки информации сильно изменилась. В области суперкомпьютерных вычислений большое распространение получили гибридные технологии. Под гибридностью понимается наличие на узлах системы ускорителей вычислений различного рода. В качестве подобных ускорителей могут использоваться как вычислители CUDA [4] или Xeon Phi [5], так и наборы перепрограммируемых FPGA-сопроцессоров. Вышеперечисленные технологии обладают высокой степенью параллелизма и решают многие задачи гораздо эффективнее классических CPU-устройств. В основном гибридные технологии применяются для выполнения физических расчетов, химического моделирования и т. д. Однако проводятся исследования возможности применения GPGPU/FPGA-устройств в качестве криптографических сопроцессоров, модулей сжатия/распаковки, подсистем предварительной обработки графической информации и т. д. Существует реальная потребность в высокопроизводительных версиях криптографических алгоритмов для применения их в гибридных суперкомпьютерных системах сбора и обработки информации. При разработке новых алгоритмов генерации ПСЧ (либо реализации старых) для параллельных систем следует учитывать специфику работы с памятью, согласно которой множественные операции с быстрой памятью обходятся дешевле, чем одиночные операции при работе с медленной.

Связанные работы

Начиная с момента своего появления технологии разработки для гибридных систем (такие как NVIDIA CUDA и AMD APP) получили широкий отклик в научной среде. Значительное число работ связано с вопросами повышения производительности алгоритмов криптозащиты с использованием технологии CUDA. В работе [6] авторы анализируют преимущества технологии CUDA в сравнении с ранее использовавшейся OpenGL. Согласно логике работы, реализованной исследователями параллельной версии алгоритма AES, входная информация хранилась в глобальной памяти CUDA-устройства, а блоки замен были помещены в область кэшируемой константной памяти для ускорения процесса чтения данных из S-блока.

Авторы исследования [7] в своей работе выполнили обзор нескольких возможных подходов к реализации многопоточной версии AES. В статье приводятся результаты сравнения производительности последовательной, параллельной CPU- и GPU-версий. Как результат, самой производительной оказалась GPU-реализация.

В статье [8] авторы приводят обзор всех режимов шифрования AES и выделяют пригодные для распараллеливания: ECB (Electronic codebook), CTR (Counter). Для режима CBC (Cipher-block chaining) возможно выполнить распараллеливание только операции расшифрования. Для данных режимов авторы приводят графики пропускной способности и некоторые советы по повышению производительности.

Помимо статей, посвященных реализации алгоритма AES на CUDA, существует также масса работ, описывающих реализацию параллельных версий прочих алгоритмов шифрования на GPGPU-системах. Вследствие своей архитектуры (квадрат) именно AES обладает одной из наилучших способностей к распараллеливанию и обеспечивает высокую криптостойкость результатов шифрования.

Общие вопросы реализации алгоритмов генерации ПСЧ на CUDA-системах

Во время реализации алгоритмов шифрования на CUDA серьезную роль играет значение гранулярности при распределении открытого текста по нитям. Очевидно, что при гранулярности, равной размеру блока, каждая нить шифрует свою часть сообщения независимо от других. При гранулярности меньшей, чем размер блока, возникает необходимость синхронизации одновременно работающих нитей. В статье [9] авторы анализируют возможность применения этих подходов для организации параллельных операций AES-шифрования. Подход, при котором каждая нить шифрует свой фрагмент открытого текста (блок AES — 16 байт), оказался наиболее выигрышным.

В гибридных системах пересылка данных между основной памятью (host memory) и памятью CUDA-устройства (device memory) по шине PCI-E может занимать некоторое время. Чтобы избежать потерь производительности, операции пересылки и расчетов обычно выполняют по конвейерному принципу.

Также при реализации любого алгоритма шифрования на GPU нужно в полной мере задействовать особенности архитектуры CUDA-систем. В наибольшей степени это относится к работе с иерархией памяти, схематичное изображение которой представлено на рис. 1.

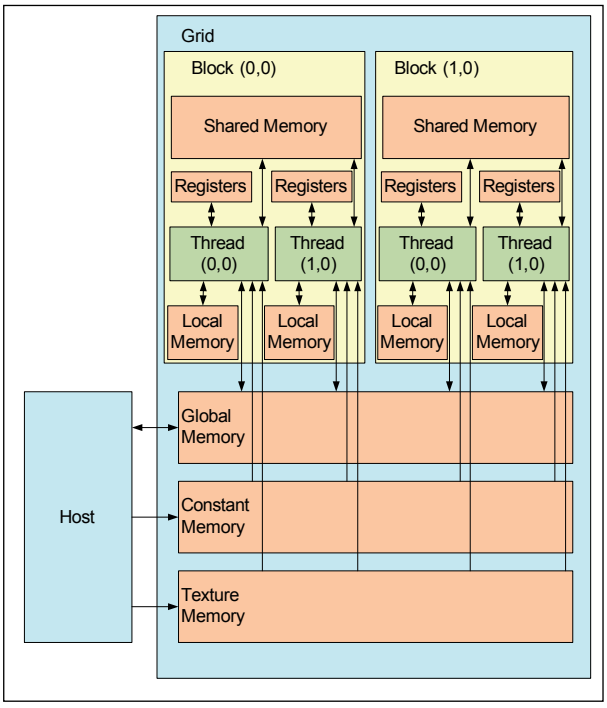

Рис. 1. Иерархия памяти CUDA

Наиболее медленной является глобальная память (global memory), поэтому перед обработкой блока данных для шифрования его сначала загружают в начиповую память (on-chip memory), в качестве которой могут выступать: разделяемая память (shared memory) или регистры (registers). Неменяющиеся данные (например, блоки замен) часто размещаются в константной памяти (constant memory), которая кэшируется процессором. К разделяемой памяти имеют доступ все нити в

пределах блока (block). После окончания очередного этапа преобразований данных в разделяемой памяти необходимо выполнять барьерную синхронизацию (вызов метода __syncthreads();).

Способ реализации алгоритма DOZEN для гибридной CUDA-системы

Блочный алгоритм DOZEN [10, 11] обладает высокой степенью параллелизма: вся поступающая информация делится на блоки по 512 бит, представляющие собой кубы 4 x 4 x 4 байт, где каждый полученный куб преобразуется по «слоям» в трех направлениях: X, Y и Z. Слой представляется в виде двумерного массива в 16 байт (4 x 4). Модификации каждого слоя-квадрата происходят на основе базовых функций стандарта AES с заменой операции сдвига строк на операцию их перемешивания. Благодаря такой схеме полное рассеивание происходит быстрее.

Алгоритм преобразования DOZEN выглядит следующим образом:

- Входной блок проходит операцию сложения по модулю 2 каждого элемента с соответствующим ему элементом раундового ключа.
- Блок расщепляется на слои по направлению оси X, над каждым слоем выполняются операции преобразования слоя.
- Блок расщепляется на слои по направлению оси Y, над каждым слоем выполняются операции преобразования слоя.
- Блок расщепляется на слои по направлению оси Z, над каждым слоем выполняются операции преобразования слоя.

Операции преобразования алгоритма DOZEN:

- SubBytes — замена байт;
- MixRows — перемешивание строк, аналогична MixColumns;
- MixColumns — полностью повторяет аналогичное преобразование AES;
- AddRoundSubKey — сложение с раундовым ключом.

Для реализации алгоритма DOZEN на CUDA была выбрана гранулярность 16 байт на нить (каждая нить обрабатывает свой слой). Перед переразбиением блока DOZEN на слои происходит синхронизация нитей.

Данные отправляются на устройство в виде одномерного массива байт. На рис. 2 представлена схема соответствия индексов одномерного массива ячейкам куба (разбиение по оси X).

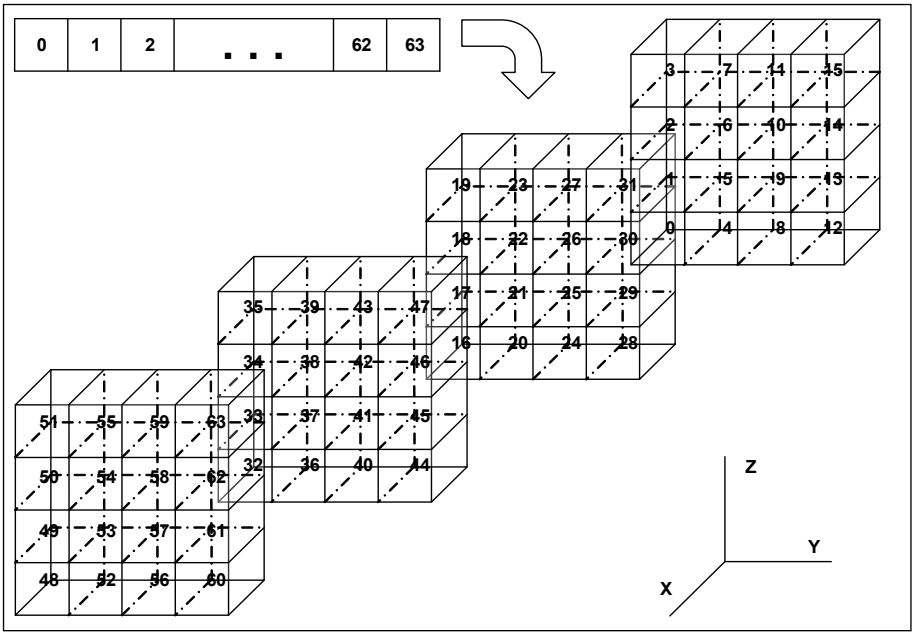

Рис. 2. Соответствие индексов массива ячейкам куба DOZEN

Для уменьшения нагрузки на вычислительные блоки CUDA-ядер индексная арифметика максимально упрощена, выполнено также «разворачивание» циклов.

Блок замен размещается в кэшируемой константной памяти. В качестве эксперимента выполнялась генерация псевдослучайных чисел по алгоритму DOZEN. Каждый блок данных (512 бит) инициализировался значением счетчика. Сформированный массив пересылался на устройство. Нить выполняла загрузку данных из глобальной памяти в разделяемую. Некоторые данные (например, линейки байт для операций перемешивания) размещались в регистровой памяти для повышения производительности. Схема алгоритма параллельной CUDA-версии алгоритма DOZEN представлена на рис. 3.

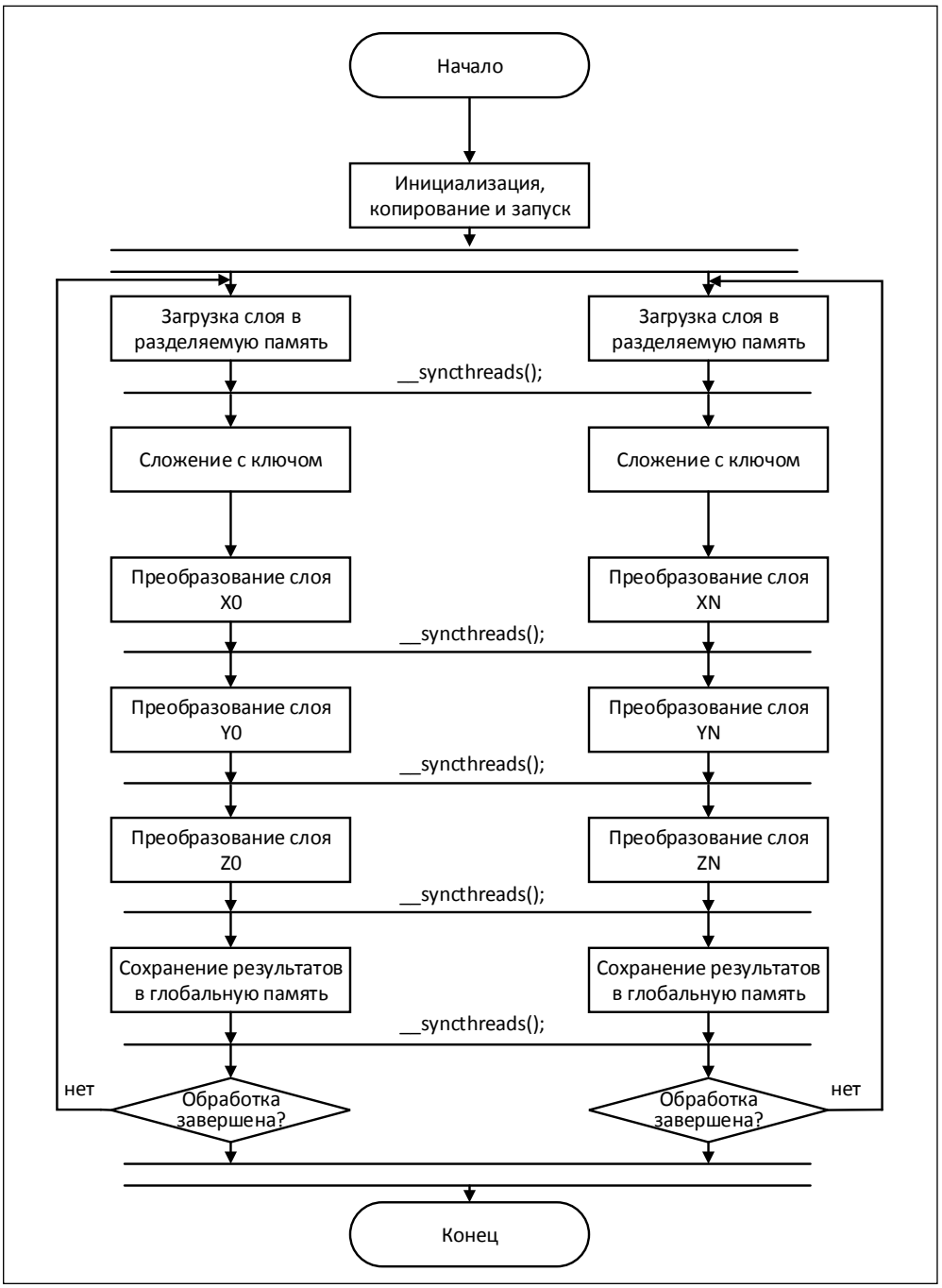

Рис. 3. Схема параллельной CUDA-версии алгоритма DOZEN

Результаты

График зависимости времени генерации последовательности от размера последовательности представлен на рис. 4. Для сравнения приведена зависимость для последовательной реализации алгоритма. Как можно заметить, GPU-версия более чем в 30 раз производительнее последовательной CPU-версии. На генерацию 1 Гб данных у GPU-версии ушло порядка 4 секунд, что соответствует пропускной способности 2 Гбит/с.

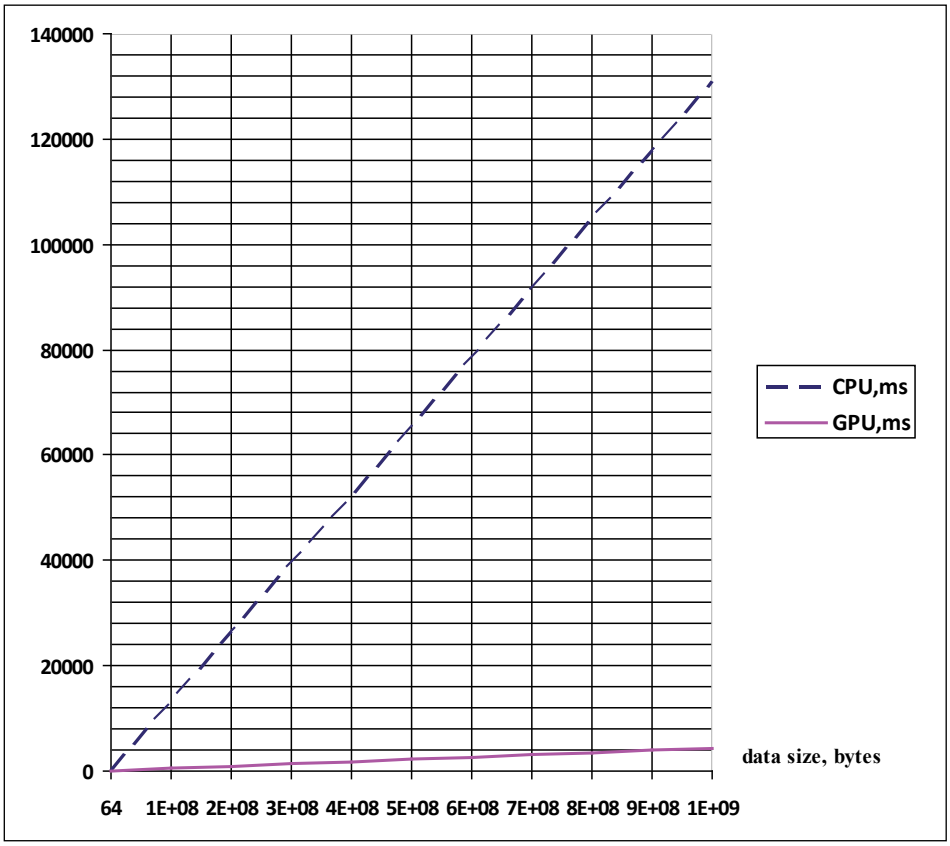

Рис. 4. График зависимости времени генерации от размера последовательности

Данные результаты были получены с использованием ускорителя GTX 680, архитектура которого соответствует урезанной версии ускорителя K20X, использующегося в суперкомпьютере Titan Cray XK7, расположенном в Национальной лаборатории Окриджа и занявшем первое место в 40-м рейтинге TOP500 (ноябрь 2012 г.) [12].

Несмотря на все возрастающую вычислительную мощь современных гибридных систем, комплексы на жесткой логике были и остаются самыми производительными и существенно превосходят микропроцессорные решения.

Способы аппаратной реализации алгоритма DOZEN

Алгоритм DOZEN удобен не только для программной, но и для аппаратной реализации. Поскольку он является алгоритмом с высокой степенью параллелизма в обработке данных, существует возможность значительно повысить его производительность.

Аппаратную реализацию алгоритма можно осуществить в двух формах — в виде специализированной интегральной микросхемы или на базе программируемой интегральной микросхемы — FPGA (Field-Programmable Gate Array). В данной статье рассмотрен второй способ — реализация на базе FPGA. В качестве тестового устройства выбрана отладочная плата с кристаллом FPGA фирмы Xilinx семейства Spatran 6 — XC6SLX100-3FGG484.

Для оценки эффективности различных способов реализаций алгоритма необходимо определить критерии, по которым их можно сравнить. Одним из основных критериев может выступать пропускная способность, определяющая объем обрабатываемой информации в единицу времени. Второй немаловажный параметр, характеризующий любую аппаратную реализацию алгоритма, — это потребляемые ресурсы. При реализации на FPGA в роле такого ресурса выступают таблицы замен (Look-Up Table, LUT) и триггеры.

Для начала рассмотрим базовый способ реализации алгоритма, который будет выступать отправной точкой. На рис. 5а приведена архитектура самого простого варианта реализации алгоритма.

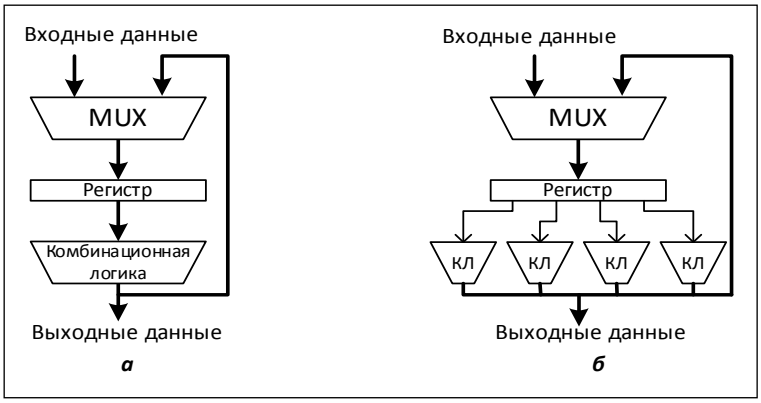

Рис. 5. Реализация алгоритма: а — базовая; б — с параллельной обработкой слоев

Данная реализация представляет собой регистр для хранения текущего состояния с мультиплексорами на входе и на выходе и комбинационную логику (КЛ), реализующую все преобразования каждого из шагов алгоритма. При использовании подобной архитектуры за один такт обрабатывается только один слой куба.

Ввиду того что параллельные слои обрабатываются независимо, можно одновременно обрабатывать по четыре слоя. Этот вариант архитектуры приведен на рис. 5б. При использовании такой архитектуры работа алгоритма ускорится в четыре раза, и на все преобразования потребуется всего три такта.

Говоря об оптимизации алгоритма, следует отметить тот факт, что алгоритм DOZEN был разработан на базе двухмерного алгоритма Rijndael, что частично дает возможность применить к нему способы оптимизации этого алгоритма. Два способа оптимизации рассмотрены в статье [13]. Первый из них — разворачивание циклов. Идея этого метода проиллюстрирована рис. 6. Данные не записываются назад во входной регистр и идут сразу в следующую комбинационную схему.

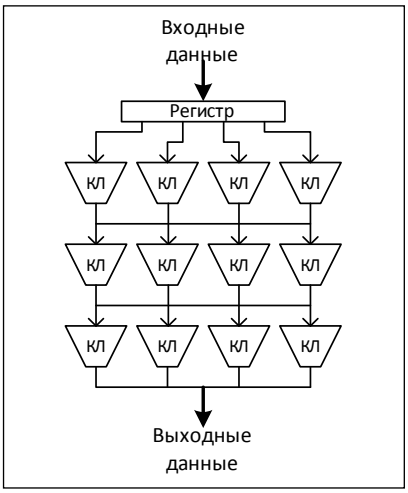

Рис. 6. Разворачивание циклов алгоритма DOZEN

При этом если в алгоритме Rijndael возможно частичное разворачивание циклов [14], то есть вместо десяти циклов можно ввести два цикла по пять преобразований или пять циклов по два, то для алгоритма DOZEN, который имеет всего три цикла, можно выполнить только полное разворачивание. И в таком случае можно сказать, что увеличение быстродействия будет незначительным, в отличие от увеличения числа потребляемых ресурсов.

Второй способ повышения производительности — это использование конвейерной обработки данных. Построение конвейера происходит путем добавления промежуточных регистров. В этом случае можно пойти двумя путями:

- добавление регистров между преобразованиями слоев;
- добавление регистров в схемы преобразования слоев.

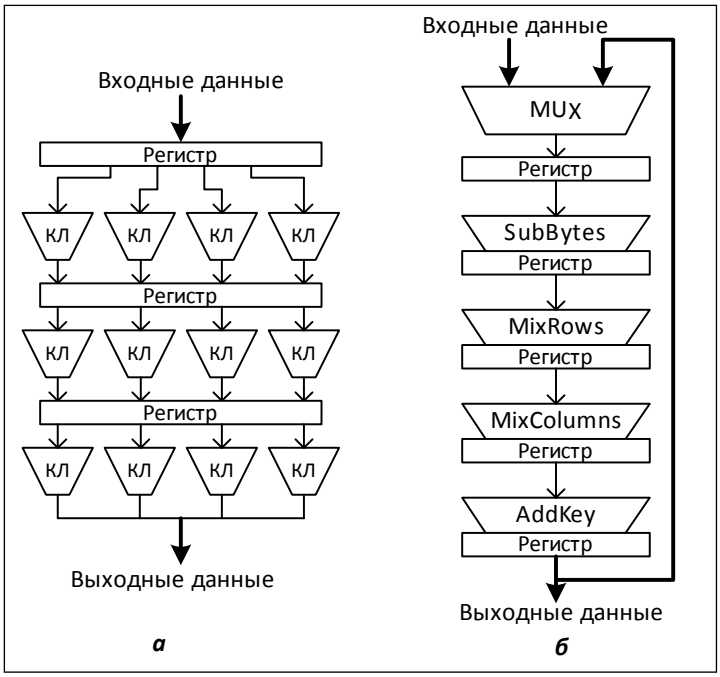

Рис. 7. Конвейерная реализация:
а — на уровне обработки кубов данных;
б — на уровне обработки одного слоя

Рассмотрим эти способы более детально. На рис. 7а приведена архитектура с добавлением промежуточных регистров между обработкой слоев. При таком подходе конвейер будет иметь три ступени. При этом, так как вторая и третья ступени конвейера идентичны и выполняются за одно время, в состав первой ступени входит также процедура предварительного сложения с ключом, что увеличивает задержку ее работы. Можно вынести эту операцию в отдельную ступень конвейера, что позволит увеличить частоту синхронизации конвейера в целом.

Второй вариант формирования конвейера — разделение каждого преобразования слоя на отдельные ступени. Рис. 7б иллюстрирует данный вариант архитектуры. Каждое преобразование слоя содержит в себе четыре шага: SubBytes, MixRows, MixColumns и AddRoundKey. Наиболее логичным и простым является решение, при котором регистры размещаются между выполнениями этих операций. Таким образом, можно получить две или четыре ступени конвейера.

Можно также комбинировать способы построения конвейеров. На рис. 8 приведена архитектура, использующая как конвейер внутри каждого преобразования слоя, так и конвейер из этих конвейеров для глобального преобразования.

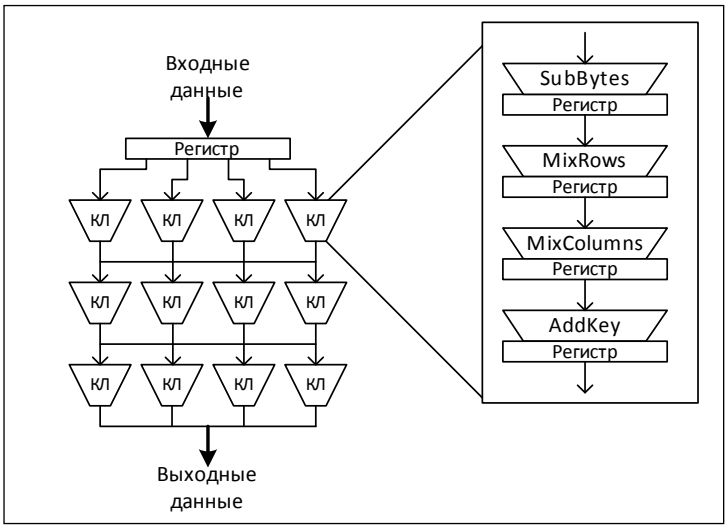

Рис 8. Совмещенный конвейер

Оценка эффективности вариантов реализации на FPGA

Для оценки эффективности предложенных архитектур они были реализованы для системы с кристаллом FPGA Xilinx Spatran 6 – XC6SLX100-3FGG484 [15]. Оценка максимальной частоты работы схемы и потребляемых ресурсов проводилась с помощью средств САПР Xilinx ISE.

Реализованы три варианта архитектуры: один — с конвейером на уровне обработки кубов данных и два варианта с конвейером на уровне обработки отдельных слоев (рис. 8) — с двумя и с четырьмя ступенями на слой соответственно.

В таблице 1 представлены результаты тестирования архитектур.

Таблица 1. Сравнение способов реализации алгоритма DOZEN по критериям

Архитектура	Логические ресурсы (LUT-ы)	Ресурсы памяти (триггеры)	Максимальная частота, МГц	Максимальная пропускная способность алгоритма, Гбит/с
Базовая реализация	6032	1123	110	18,773
Конвейер уровня кубов	11072	2047	105	53,760
Двухступенчатый конвейер уровня слоев	10844	3584	205	104,960
Четырехступенчатый конвейер уровня слоев	12360	6655	280	143,360

На графике (рис. 9) приведена зависимость пропускной способности от потребляемых ресурсов.

На фоне полученных результатов возникает проблема интерфейса для вывода полученных данных. Существующие в настоящее время интерфейсы для FPGA позволяют выводить данные на скорости не более 100 Гбит в секунду (в FPGA Virtex UltraScale, анонсированной фирмой Xilinx в 2013 г., имеется аппаратная реализация блока 100 G Ethernet, который можно сконфигурировать как 10 x 10 G или 4 x 25 G).

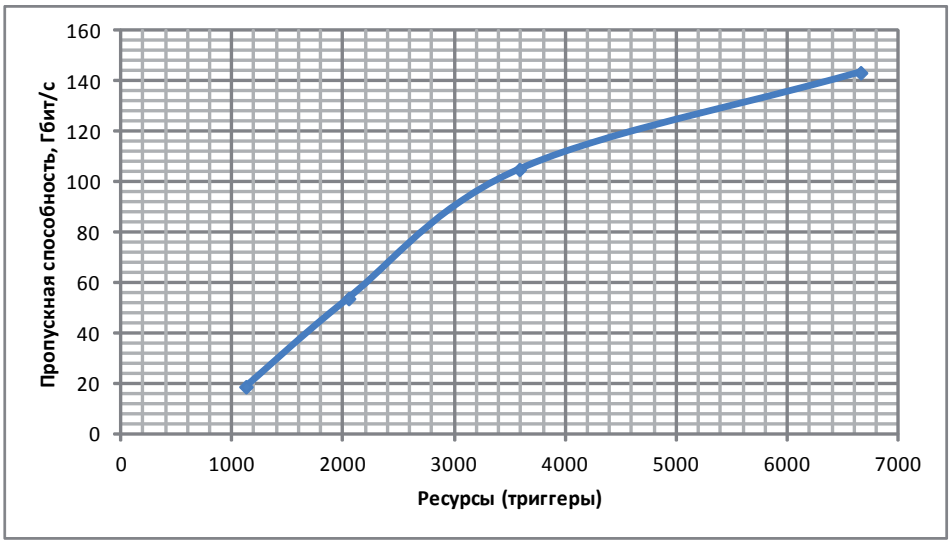

Рис. 9. Зависимость пропускной способности от потребляемых ресурсов

Заключение

Представленные в статье результаты подтверждают целесообразность использования многомерных алгоритмов стохастического преобразования в высокопроизводительных промышленных GPGPU-системах. За счет высокой степени параллелизма алгоритм DOZEN обладает высокой пропускной способностью в многопоточных системах, что является особенно актуальным по причине массовой переориентации основных вендоров оборудования для ЦОД на гибридные решения. Для систем специального назначения, применяемых на контролируемом оборудовании АСУ ТП, по-прежнему эффективным решением является аппаратный подход к реализации алгоритма DOZEN.

СПИСОК ЛИТЕРАТУРЫ:

1. SCADA — Портал по информационной безопасности SecurityLab. URL: http://www.securitylab.ru/news/tags/SCADA/ (дата обращения: 25.05.2014).
2. Security Guidance — Портал Cloud Security Alliance. URL: https://downloads.cloudsecurityalliance.org/initiatives/guidance/csaguide.v3.0.pdf (дата обращения: 25.05.2014).
3. Безопасность центров обработки данных — Seagate официальный сайт. URL: http://www.seagate.com/ru/ru/solutions/security/data-center-security/ (дата обращения: 25.05.2014).
4. *Боресков А. В. и др.* Параллельные вычисления на GPU. Архитектура и программная модель CUDA. М.: Издательство МГУ, 2013. — 335 с.
5. Intel Developer Zone: Intel Xeon Phi Coprocessor — Официальный портал Intel URL: http://software.intel.com/mic-developer (дата обращения: 25.05.2014).
6. *Manavski S. A.* CUDA compatible GPU as an efficient hardware accelerator for AES cryptography // ICSPC 2007 IEEE Los Alamitos. November 2007. P. 65–68.
7. *Ortega J., Trefftz H., Trefftz C.* «Parallelizing AES on multicores and GPUs» in IEEE International Conference on Electro/Information Technology (EIT), Mankato, MN, 2011. P. 1–5.
8. *Qinjian L. et al.* Implementation and Analysis of AES Encryption on GPU // Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems. 2012. June 25–27. P. 843–848.
9. *Iwai K., Nishikawa N., Kurokawa T.* Acceleration of AES encryption on CUDA GPU // International Journal of Networking and Computing. 2012. № 1 (2). P. 131–145.
10. *Иванов М. А., Васильев Н. П., Чугунков И. В. и др.* Трехмерный генератор псевдослучайных чисел, ориентированный на реализацию в гибридных вычислительных системах // Вестник НИЯУ МИФИ. 2012. № 2. С. 232–235.

11. *Иванов М. А., Васильев Н. П., Чугунков И. В. и др.* Способ нелинейного трехмерного многораундового преобразования данных DOZEN. Патент RU 2 503 994 C1. 2012.
12. List of Top500 supercomputers — Официальный портал. URL: http://top500.org (дата обращения: 25.05.2014).
13. *Gaj K., Chodowiec P.* FPGA and ASIC Implementations of AES // Cryptographic Engineering. 2009. P. 235–294.
14. *Zhang X., Parhi K. K.* High-Speed VLSI Architectures for the AES Algorithm // IEEE transactions on very large scale integration (VLSI) systems. Vol. 12. № 9. 2004. P. 957 – 967.
15. Spartan-6 FPGA Family — Официальный портал Xilinx. URL: http://www.xilinx.com/products/silicon-devices/fpga/spartan-6/ (дата обращения: 25.05.2014).

REFERENCES:

1. SCADA — Portal of Information Security. SecurityLab. URL: http://www.securitylab.ru/news/tags/SCADA/ (accessed: 25.05.2014).
2. Security Guidance — Cloud Security Alliance Portal. URL: https://downloads.cloudsecurityalliance.org/initiatives/guidance/csaguide.v3.0.pdf (accessed: 25.05.2014).
3. Security of data centers — Seagate official site. URL: http://www.seagate.com/ru/ru/solutions/security/data-center-security/ (accessed: 25.05.2014).
4. *Boreskov A. V. et al.* Parallel computing on the GPU. Architecture and programming model of CUDA. Moscow: MSU publishing house, 2013. — 335 p.
5. Intel Developer Zone: Intel Xeon Phi Coprocessor — Intel official site. URL: http://software.intel.com/mic-developer (accessed: 25.05.2014).
6. *Manavski S. A.* CUDA compatible GPU as an efficient hardware accelerator for AES cryptography // ICSPC 2007 IEEE Los Alamitos. November 2007. P. 65–68.
7. *Ortega J., Trefftz H., Trefftz C.* «Parallelizing AES on multicores and GPUs» in IEEE International Conference on Electro/Information Technology (EIT), Mankato, MN, 2011. P. 1–5.
8. *Qinjian L. et al.* Implementation and Analysis of AES Encryption on GPU // Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems. June 25–27, 2012. P. 843–848.
9. *Iwai K., Nishikawa N., Kurokawa T.* Acceleration of AES encryption on CUDA GPU // International Journal of Networking and Computing. 2012. № 1 (2). P. 131–145.
10. *Ivanov M. A., Vasilev N. P., Chugunkov I. V. et al.* A three-dimensional pseudo-random number generator, focused on the implementation in hybrid computing systems // Bulletin of MEPhI. 2012. № 2. P. 232–235
11. *Ivanov M. A., Vasilev N. P., Chugunkov I. V. et al.* Nonlinear method of three-dimensional multi-round data transformation DOZEN, Patent RU 2 503 994 C1. 2012.
12. List of Top500 supercomputers — official site. URL: http://top500.org (accessed: 25.05.2014).
13. *Gaj K., Chodowiec P.* FPGA and ASIC Implementations of AES // Cryptographic Engineering. 2009. P. 235–294.
14. *Zhang X., Parhi K. K.* High-Speed VLSI Architectures for the AES Algorithm // IEEE transactions on very large scale integration (VLSI) systems. Vol. 12. № 9. 2004. P. 957 – 967.
15. Spartan-6 FPGA Family — Xilinx official site. URL: http://www.xilinx.com/products/silicon-devices/fpga/spartan-6/ (accessed: 25.05.2014).