Наука и Образование
МГТУ им. Н.Э. Баумана
Сетевое научное издание
ISSN 1994-0408

Наука и Образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2016. № 10. С. 132–142.

DOI: **10.7463/1016.0847891**

Представлена в редакцию: 14.09.2016
Исправлена: 28.09.2016

УДК 004.056.55

Производительность древовидных криптографических хэш-функций, основанных на клеточных автоматах, при их реализации на графических процессорах

Ключарёв П. Г.[1,*]

[*]pk.iu8@yandex.ru

[1]МГТУ им. Н.Э. Баумана, Москва, Россия

Статья посвящена тестированию производительности криптографических хэш-функций, основанных на обобщенных клеточных автоматах и имеющих древовидную схему построения, при программной реализации на графических процессорах фирм AMD и NVIDIA. Реализация производилась с использованием интерфейса OpenCL. Производительность полученной реализации составила от 700 до 3500 Мбит/с, в зависимости от используемого графического процессора и ряда параметров алгоритма, что является хорошим результатом, учитывая, что рассматриваемые хэш-функции, как и другие криптоалгоритмы, основанные на обобщенных клеточных автоматах, предназначены для аппаратной реализации. Возможность достижение такого уровня производительности для программной реализации существенно расширяет область применения данных хэш-функций. Работа выполнена при поддержке РФФИ, проект №16-07-00542.

Ключевые слова: клеточный автомат, хэш-функция, графический процессор

Введение

Обеспечение информационной безопасности в настоящее время является важнейшей задачей, возникающей в процессе проектирования и эксплуатации информационных систем различного назначения. Для решения этой задачи часто используются различные криптографические алгоритмы, в том числе, криптографические хэш-функции. При этом требования к скорости обработки информации постоянно возрастают. Это приводит к необходимости разработки высокопроизводительных криптографических хэш-функций. Разработанные автором методы синтеза высокопроизводительных криптографических хэш-функций, основанные на использовании обобщенных клеточных автоматов, дают возможность производить построение хэш-функций, показывающих высокую производительность аппаратной реализации. Реализация же таких хэш-функций на обычных микропроцессорах не демонстрирует высокой производительности. Этот факт не является недостатком, а показывает сферу применимости таких хэш-функций.

Эта статья продолжает серию статей ([4; 5; 6; 7; 9; 10] и др.), посвященных криптографическим алгоритмам, основанным на обобщенных клеточных автоматах, методам их построения и реализации. Целью данной статьи является исследование возможности реализации рассматриваемых хэш-функций на графических процессорах и тестирование производительности такой реализации. В данной статье показывается, что программная реализация рассматриваемых хэш-функций на графических процессорах показывает достаточно высокий уровень производительности, что существенно расширяет сферу применения данных хэш-функций.

Графические процессоры

Графические процессоры (GPU) изначально создавались для ускорения трехмерной графики, однако впоследствии стало ясно, что с их помощью можно производить высокоскоростные параллельные вычисления, т.к. они работают как большое количество специализированных процессоров. Такие вычисления могут быть востребованы в различных областях знаний. Им посвящено большое количество источников, например, [13; 14; 15; 16; 17].

В настоящее время, графические процессоры обладают высокой производительностью при решении ряда задач (до 1 терафлопса). Программы для графических процессоров разрабатывают с помощью специальных API. Наиболее часто используемыми API являются OpenCL и CUDA. Программа, использующая эти API, состоит из ядра, которое выполняется на графическом процессоре и хост-программы, которая исполняется на центральном процессоре.

После того, как хост-программа начинает выполнять ядро, определяется так называемое индексное пространство, для каждой точки которого выполняется экземпляр ядра, называемый рабочим элементом. Различные рабочие элементы различаются значениями локального и глобального идентификаторов. Рабочие элементы объединяются в рабочие группы, каждая из которых также имеет свой уникальный идентификатор. При этом сочетание локального идентификатора и идентификактора рабочей группы однозначно определяет рабочий элемент.

Размер рабочей группы ограничен. Это ограничение свое для каждого устройства. Так, для графических процессоров фирмы AMD этот параметр равен 256. Для графических процессоров фирмы NVIDIA данный параметр зависит от версии CUDA. Для версий 1.1 - 1.3 он равен 512, а для более поздних – 1024. Таким образом, при реализации на графических процессорах производства NVIDIA, максимальный размер графа равен 1024.

Обобщенные клеточные автоматы

Весьма перспективным криптографическим примитивом является обобщенный клеточный автомат. Кратко напомним его определение.

Будем называть *обобщённым клеточным автоматом* ориентированный мультиграф $A(V,E)$, где $V=\{v_1,\ldots,v_N\}$ - множество вершин, а E -мультимножество ребер. С каждой вершиной v_i этого графа ассоциированы:

- булева переменная m_i, которая называется *ячейкой*;
- булева функция $f_i(x_1,\ldots,x_{d_i})$, которая называется локальной функцией связи i-й вершины.

При этом каждой паре (v,e), где v - вершина, а e - инцидентное ей ребро, будет соответствовать номер аргумента локальной функции связи, вычисляемой в вершине v (номер ребра e относительно вершины v).

Обобщенный клеточный автомат работает по шагам. Перед первым шагом каждая ячейка m_i, $i=1...N$, имеет начальное значение $m_i(0)\in\{0,1\}$. Далее, значения ячеек на шаге t вычисляются по формуле:

$$m_i(t)=f_i(m_{\eta(i,1)}(t-1),m_{\eta(i,2)}(t-1),\ldots,m_{\eta(i,d_i)}(t-1)), \qquad (1)$$

где $\eta(i,j)$ – номер вершины, из которой исходит ребро, заходящее в вершину i и имеющее относительно этой вершины номер j. Заполнением клеточного автомата на шаге t будем называть набор значений ячеек $(m_1(t),m_2(t),\ldots,m_N(t))$.

Обобщенный клеточный автомат будем называть *однородным*, если локальная функция связи для всех ячеек одинакова. Назовем обобщённый клеточный автомат *неориентированным*, если в графе для любого ребра (u,v) в существует и ребро (v,u). Граф такого автомата можно рассматривать как неориентированный, если заменить каждую пару ориентированных ребер (u,v) и (v,u) на неориентированное ребро $\{u,v\}$.

Здесь мы будем использовать лишь неориентированные однородные обобщённые клеточные автоматы, для краткости называя их просто обобщёнными клеточными автоматами.

Пусть $F_t:\{0,1\}^n\to\{0,1\}^n$ - функция, аргументом которой является начальное заполнение данного обобщенного клеточного автомата, а значением – заполнение этого автомата через *t* шагов.

Хэш-функции

Здесь мы лишь очень кратко остановимся на структуре реализуемых хэш-функций. Подробную информацию о них можно найти в статье [4].

Хэш-функция основана на обобщенных клеточных автоматах. Будем использовать функции вида $S_c:\{0,1\}^k\times\{0,1\}^n\to\{0,1\}^m$, которые основываются на обобщённых клеточных автоматах и могут быть заданы формулой

$$S_c^A(key, x) = pr_m(F_A(x \| key \| c, r)),$$

где $x \| y$ - конкатенация x и y,

r – число шагов клеточного автомата,

$pr_m : B^* \to B^m$ – функция, возвращающая младшие m элементов аргумента;

A – обобщённый клеточный автомат;

$c \in B^t$ – некоторая константа, вес которой близок к значению $[t/2]$.

Константа c необходима для улучшения лавинного эффекта и обеспечения отсутствия неподвижных точек. Число шагов клеточного автомата r и число скрытых вершин t выбираются так, чтобы функцию нельзя было отличить от случайной при помощи статистических тестов (см. работу работе [8]).

Автором в работе [8] было показано, что такие функции являются неотличимыми от псевдослучайных с помощью стандартного набора статистических тестов NIST, в случае правильного выбора параметров, в том числе графа клеточного автомата и локальной функции связи.

Разобьем сообщение X на блоки: $X = (x_1, x_2, \ldots, x_\eta)$ (если сообщение не кратно длине блока, дополним его до длины блока). Далее, мы воспользуемся тем, что функция S_c^A в случае правильного выбора параметров представляет собой семейство однонаправленных псевдослучайных функций и сформируем хэш-функцию следующим образом:

$$H(X) = pr_s\left(S_{c_2}^A\left(c_3, \oplus_{i=1}^{\eta} S_{c_1}^A(i, x_i)\right)\right),$$

где $c_1, c_2 \in B^t$ - различные константы, вес которых близок к значению $[t/2]$;

$c_3 \in B^k$ – константа, вес которой близок к значению $[k/2]$;

t – число скрытых вершин графа.

s – длина хэша.

Реализация

Хэш-функции были реализованы на графических процессорах с использованием такого универсального API как OpenCL. Программа была написана на C++ и позволяла пользователю устанавливать различные параметры, в том числе граф клеточного автомата, локальную функцию связи, константы и т.д.

Как известно, обобщенный клеточный автомат состоит из набора ячеек, над которыми производятся однотипные вычисления. Это дает возможность эффективной реализации на графических процессорах, так как процесс таких вычислений легко распараллеливается.

Для каждого обобщенного клеточного автомата, входящего в схему вычисления хэш-функции использовалась отдельная рабочая группа (workgroup) графического процессора. Поразрядное сложение по модулю 2 выходов обобщенных клеточных автоматов также осуществлялось с помощью графического процессора.

Размер рабочей группы выбирался кратным размеру волнового фронта графического процессора, что является одной из особенностей графических процессоров.

Использовались обобщенные клеточные автоматы с графами Любоцкого-Филипса-Сарнака, размера 242 (диаметра 6) и размер блока 160. Одновременно на графическом процессоре выполнялось до 256 автоматов.

Тестирование производительности

При проведении тестирования использовалось видеоадаптеры, основанные на следующих графических процессорах:

- NVIDIA GTX 650;
- NVIDIA GTX 770;
- AMD R9 280X.

Основные параметры этих видеоадаптеров приведены в таблице 1.

Таблица 1. Параметры графических процессоров

Параметр	**Видеокарта**		
	NVIDIA GTX 650	**NVIDIA GTX 770**	**AMD R9 280X**
Количество вычислительных элементов(compute units)	4	8	32
Тактовая частота, МГц	1033	1137	1000
Максимальный размер рабочей группы	1024	1024	256
Размер глобальной памяти, МБ	1024	2048	2048
Размер локальной памяти, КБ	48	48	32
Тип памяти	GDDR5	GDDR5	GDDR5
Год появления на рынке	2012	2013	2014

Результаты произведенного тестирования производительности для различных значений числа шагов *r* приведены в табл. 2 и на рис. 1.

Таблица 2. Производительность хэш-функций

Число шагов *r*	Производительность, Мбит/сек		
	NVIDIA GTX650	NVIDIA GTX770	AMD R9 280X
5	1003	2212	3523
6	923	1978	3001
7	805	1728	2957
8	742	1591	2657
9	712	1512	2409

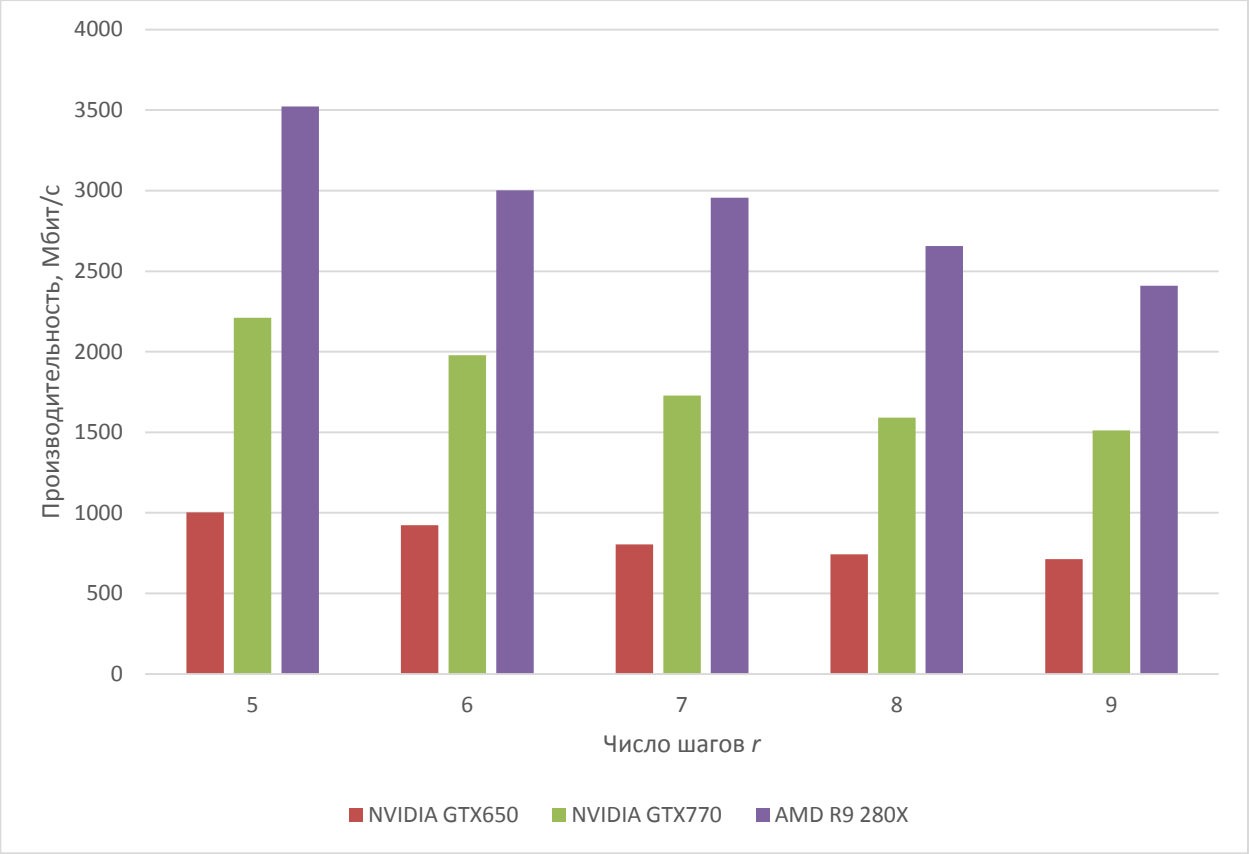

Рис. 1 – производительность хэш-функции.

Обсуждение результатов

Обобщенные клеточные автоматы являются криптографическими примитивами, рассчитанными на аппаратную реализацию. Основанные на них криптографические алгоритмы показывают высокую производительность, будучи реализованы аппаратно (например, на базе ПЛИС). В то же время, производительность их программной реализации весьма невысока. Данные, приведенные в предыдущем разделе, убедительно свидетельствуют, что применение графических процессоров позволяет существенно ускорить работу криптографических хэш-функций, основанных на обобщенных клеточных автоматах. Учитывая, что современные компьютеры, смартфоны и планшеты как правило снабжены

графическими процессорами, этот факт существенно расширяет область применения этих хэш-функций.

Рассматриваемые хэш-функции могут быть применены в различных приложениях, в частности в системах электронной подписи [12], системах организации доступа к данным [1; 3; 11] и других задачах информационной безопасности (например, упомянутых в работе [2]).

Заключение

Таким образом была продемонстрировано, что предложенное автором в работе [4] семейство древовидных хэш-функций, основанных на обобщенных клеточных автоматах, допускает достаточно эффективную реализацию на графических процессорах.

Работа выполнена при поддержке РФФИ, проект №16-07-00542.

Список литературы

1. Быков А.Ю. Алгоритмы распределения ресурсов для защиты информации между объектами информационной системы на основе игровой модели и принципа равной защищенности объектов // Наука и образование. Электронное научно-техническое издание. 2015. № 9.
2. Быков А.Ю., Артамонова А.Ю. Модификация метода вектора спада для оптимизационно-имитационного подхода к задачам проектирования систем защиты информации // Наука и образование. Электронное научно-техническое издание. 2015. № 1.
3. Быков А.Ю., Панфилов Ф.А., Ховрина А.В. Алгоритм выбора классов защищенности для объектов распределенной информационной системы и размещения данных по объектам на основе приведения оптимизационной задачи к задаче теории игр с непротивоположными интересами // Наука и образование. Электронное научно-техническое издание. 2016. Т. 1.
4. Ключарев П.Г. Криптографические хэш-функции, основанные на обобщённых клеточных автоматах // Наука и образование. Электронное научно-техническое издание. 2013. № 1.
5. Ключарев П.Г. О вычислительной сложности некоторых задач на обобщенных клеточных автоматах // Наука и образование. Электронное научно-техническое издание. 2012. № 1.
6. Ключарев П.Г. О периоде обобщённых клеточных автоматов // Наука и образование. Электронное научно-техническое издание. 2012. № 2.
7. Ключарев П.Г. Обеспечение криптографических свойств обобщённых клеточных автоматов // Наука и образование. Электронное научно-техническое издание. 2012. № 3.
8. Ключарев П.Г. Построение псевдослучайных функций на основе обобщённых клеточных автоматов // Наука и образование. Электронное научно-техническое издание. 2012. № 10.

9. Ключарев П.Г. Производительность и эффективность аппаратной реализации поточных шифров, основанных на обобщенных клеточных атоматах // Наука и образование. Электронное научно-техническое издание. 2013. № 10. — С. 299-314.
10. Ключарёв П.Г. Реализация криптографических хэш-функций, основанных на обобщенных клеточных автоматах, на базе ПЛИС: производительность и эффективность // Наука и образование. Электронное научно-техническое издание. 2014. № 1.
11. Лебедев А.Н. Способ рассылки защищенных данных с регулированием доступа к отдельным их разделам // Вопросы кибербезопасности. 2015. № 5. — С. 70-72.
12. Лебедев А.Н. Электронная подпись: новый этап // Вестник Московского городского педагогического университета: серия Экономика. 2013. № 1. — С. 43-51.
13. Eberly D.H. GPGPU Programming for Games and Science. Taylor & Francis, 2014.
14. Gaster B., Howes L., Kaeli D.R., Mistry P., Schaa D. Heterogeneous Computing with OpenCL: Revised OpenCL 1.2 Edition. Elsevier Science, 2012.
15. Kaeli D.R., Mistry P., Schaa D., Zhang D.P. Heterogeneous Computing with OpenCL 2.0. Elsevier Science, 2015.
16. Kowalik J., Puźniakowski T. Using OpenCL: Programming Massively Parallel Computers. IOS Press, 2012.
17. Scarpino M. OpenCL in Action: How to Accelerate Graphics and Computation. Manning, 2012.

Science & Education of the Bauman MSTU
Electronic journal
ISSN 1994-0408

Science and Education of the Bauman MSTU, 2016, no. 10, pp. 132–142.

DOI: **10.7463/1016.0847891**

Received: 14.09.2016
Revised: 28.09.2016

The Cryptographic Tree-Like Hash Function Performance Based on the Generalized Cellular Automata in GPU Implementation

P.G. Klyucharev[1,*]

[*]pk.iu8@yandex.ru

[1]Bauman Moscow State Technical University, Moscow, Russia

Keywords: cellular automata, hash function, GPU

Author-developed methods of synthesis of high-performance cryptographic hash functions based on the generalized cellular automata allow us to build hash functions, featuring high-performance of hardware implementation. Hash functions implemented on the conventional microprocessors did not show high performance. This fact is not the shortcoming, but shows an application scope of such-hash functions.

This article is sequel to the series of articles devoted to the cellular automata-based cryptographic algorithms, methods of their construction and implementation. The aim of this article is to study the feasibility for implementation of considered hash functions on graphics processors and to test performance of such an implementation. The article shows that the software implementation of hash functions based on GPU demonstrates a sufficiently high level of performance, thus significantly expanding the scope of application of these hash functions, because cutting-edge computers, smartphones and pads, usually, use GPU.

Hash functions have been implemented on GPUs using such a universal API as OpenCL. The program is written in C ++ and allows users to set various parameters, including the graph of the cellular automata, a local function of communication, constants, etc.

As is known, the generalized cellular automata comprise a set of cells on which computations of the same type are performed. It enables efficient implementation on GPU, since the computing process is easily parallelized. A separate working group of the GPU was used for each of generalized cellular automata, included in the computation scheme of the hash function. Bitwise modulo-2 addition of outputs of generalized cellular automata is also carried out using the GPU.

We used the generalized cellular automata with Lubotzky-Phillips-Sarnak graphs, size of 242 (diameter of 6), and block size of 160.

During the test were used graphics cards based on the following GPUs: NVIDIA GTX 650; NVIDIA GTX 770; AMD R9 280X. Testing was conducted for different numbers of steps of the cellular automata.

Depending upon the particular graphics processor and the number of steps, the performance was reached within 700 - 3500 Mbit / s.

Thus, the article has demonstrated that the author-proposed family of tree-like hash functions based on generalized cellular automata allows sufficiently efficient implementation on GPUs.

This work was supported by RFBR, project №16-07-00542.

References

1. Bykov A.Iu. Algoritmy raspredeleniia resursov dlia zashchity informatsii mezhdu ob"ektami informatsionnoi sistemy na osnove igrovoi modeli i printsipa ravnoi zashchishchennosti ob"ektov. *Nauka i obrazovanie = Science and education*. Electronic scientific and technical publication. 2015. No. 9. [In Russian]
2. Bykov A.Iu., Artamonova A.Iu. Modifikatsiia metoda vektora spada dlia optimizatsionno-imitatsionnogo podkhoda k zadacham proektirovaniia sistem zashchity informatsii. *Nauka i obrazovanie = Science and education*. Electronic scientific and technical publication. 2015. No. 1. [In Russian]
3. Bykov A.Iu., Panfilov F.A., Khovrina A.V. Algoritm vybora klassov zashchishchennosti dlia ob"ektov raspredelennoi informatsionnoi sistemy i razmeshcheniia dannykh po ob"ektam na osnove privedeniia optimizatsionnoi zadachi k zadache teorii igr s neprotivopolozhnymi interesami. *Nauka i obrazovanie = Science and education*. Electronic scientific and technical publication. 2016. Vol. 1. [In Russian]
4. Kliucharev P.G. Kriptograficheskie khesh-funktsii, osnovannye na obobshchennykh kletochnykh avtomatakh. *Nauka i obrazovanie = Science and education*. Electronic scientific and technical publication. 2013. No. 1. [In Russian]
5. 5. Kliucharev P.G. O vychislitel'noi slozhnosti nekotorykh zadach na obobshchennykh kletochnykh avtomatakh. *Nauka i obrazovanie = Science and education*. Electronic scientific and technical publication. 2012. No. 1. [In Russian]
6. Kliucharev P.G. O periode obobshchennykh kletochnykh avtomatov. *Nauka i obrazovanie = Science and education*. Electronic scientific and technical publication. 2012. No. 2. [In Russian]
7. Kliucharev P.G. Obespechenie kriptograficheskikh svoistv obobshchennykh kletochnykh avtomatov. *Nauka i obrazovanie = Science and education*. Electronic scientific and technical publication. 2012. No. 3. [In Russian]
8. Kliucharev P.G. Postroenie psevdosluchainykh funktsii na osnove obobshchennykh kletochnykh avtomatov. *Nauka i obrazovanie = Science and education*. Electronic scientific and technical publication. 2012. No. 10. [In Russian]
9. Kliucharev P.G. Proizvoditel'nost' i effektivnost' apparatnoi realizatsii potochnykh shifrov, osnovannykh na obobshchennykh kletochnykh atomatakh. *Nauka i obrazovanie = Science*

and education. Electronic scientific and technical publication. 2013. No. 10. Pp. 299-314. [In Russian]
10. Kliucharev P.G. Realizatsiia kriptograficheskikh khesh-funktsii, osnovannykh na obobshchennykh kletochnykh avtomatakh, na baze PLIS: proizvoditel'nost' i effektivnost'. *Nauka i obrazovanie = Science and education*. Electronic scientific and technical publication. 2014. No. 1. [In Russian]
11. Lebedev A.N. Sposob rassylki zashchishchennykh dannykh s regulirovaniem dostupa k otdel'nym ikh razdelam // *Voprosy kiberbezopasnosti = Questions of cybersecurity*. 2015. No. 5. Pp. 70-72. [In Russian]
12. Lebedev A.N. Elektronnaia podpis': novyi etap [Electronic signature: new stage]. *Vestnik Moskovskogo gorodskogo pedagogicheskogo universiteta: seriia Ekonomika = Bulletin of Moscow City Teacher Training University: series "Economics"*. 2013. No. 1. Pp. 43-51. [In Russian]
13. 13. Eberly D.H. GPGPU Programming for Games and Science. *Taylor & Francis*, 2014.
14. Gaster B., Howes L., Kaeli D.R., Mistry P., Schaa D. Heterogeneous Computing with OpenCL: Revised OpenCL 1.2 Edition. *Elsevier Science*, 2012.
15. Kaeli D.R., Mistry P., Schaa D., Zhang D.P. Heterogeneous Computing with OpenCL 2.0. *Elsevier Science*, 2015.
16. Kowalik J., Puźniakowski T. Using OpenCL: Programming Massively Parallel Computers. *IOS Press*, 2012.
17. Scarpino M. OpenCL in Action: How to Accelerate Graphics and Computation. *Manning*, 2012.