# ADVANCED FEATURES OF NVIDIA KEPLER ARCHITECTURE AND PARALLEL COMPUTATION PLATFORM CUDA FOR DEVELOPING SCIENTIFIC COMPUTE-INTENSIVE APPLICATIONS

***V. A. Dudnik,* V. I. Kudryavtsev, S. A. Us, M. V. Shestakov***

*National Science Center "Kharkiv Institute of Physics and Technology", 61108, Kharkiv, Ukraine*

(Received January 23, 2018)

The paper describes additional features offered by new Kepler architecture of NVIDIA graphic processors, and their usage for creating high performance programs in a wide range of scientific compute-intensive applications. Recommendations are given for their use at realization of sci-tech computation algorithms by means of graphic processors. New capabilities of the parallel computation platform CUDA are also described, in particular, regarding a set of program development tool extensions for the Fortran, C and C++ languages. The extended capabilities make it possible to minimize the time of application development and to increase the programming productivity.

PACS: 89.80.+h, 89.70.+c, 01.10.Hx

## 1. INTRODUCTION

The architecture advance of NVIDIA graphic processors has been represented by a family of GK104 and GK110 graphic processors of the Kepler architecture. As the main differences of the Kepler architecture from the previous Fermi architecture, we mention the following:

- improved energy efficiency;
- new architecture of data flow processors SMX (Streaming Multiprocessor Architecture);
- dynamic parallelization of data streams;
- hyper-Q technology.

Let us consider in greater detail the above-noted capabilities of the graphic processor architecture in terms of their use in the development of programs for scientific and technical computations.

## 2. THE INCREASED ENERGY EFFICIENCY

The improvement of energy efficiency (with simultaneous increase in productivity has been gained due to the use of new memory and bus controllers, which enabled the increase in the clock rate of memory up to $6\,GHz$. This value is considerably higher than the $4.4\,GHz$ memory rate attained with the architecture of previous generation, though it is still lower than the theoretically maximum values of $7\,GHz$ predicted for GDDR5. These graphics memory parameters have made it possible to organize the operation of shader units at the same (on one frequency with as that of the graphic processor kernel. The rejection of the model with independent shader clocks, which was used in the previous NVIDIA GPUs, has permitted the manufacturers to reduce the energy consumption owing to circuitry simplification of the graphics memory data exchange control units. This is justifiable even in view of the fact that it has taken a greater number of shader kernels to attain the processing productivity of previous developments. An additional point is that the reduction in energy consumption occurs not only because the new architecture of the memory and bus controllers is more energy-saving than the previous-generation architecture. The use of the unified clock rate leads to the decrease in the operation frequency of shader units, and this, in turn, substantially reduces the energy consumption of the graphics kernel [1,2]. As a result, two Kepler shader kernels consume about 90% of the supply power required for one kernel of the previous Fermi architecture.

## 3. THE SMX ARCHITECTURE

Essential modifications in the architecture of Kepler SMX processors provide improved energy efficiency and a 3-fold higher data processing capacity due to a new innovative structure of streaming multiprocessors. Each SMX-processor of the Kepler architecture includes four stream branch ("warp") schedulers, each scheduler providing dispatching of two parallel streams of instructions. (Fig.1).

*Corresponding author. E-mail address: vladimir-1953@mail.ru

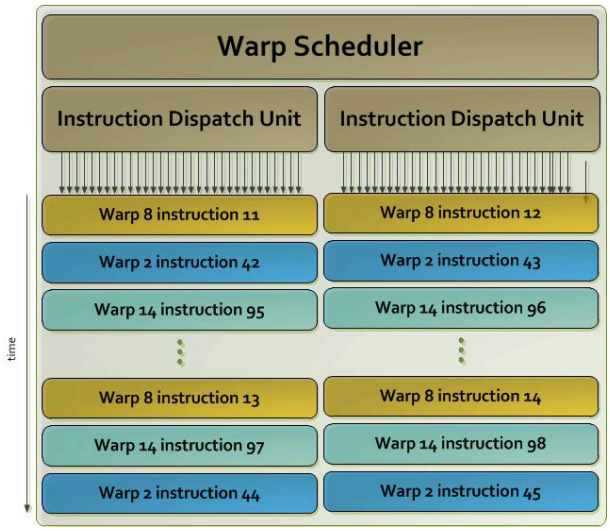


***Fig.1.*** *A single warp scheduler unit*

Thus, four complicated and energy-inefficient SMX units of the Fermi architecture are replaced by four device-simple SMX units of the Kepler architecture. This permits the arrangement of a greater number of SMX processors on the same chip area. Besides, the saving in area occupied by the control logic of the graphics processor has made it possible to use the free chip area for placing additional SMX processors (Fig.2). The complete Kepler GK110 architecture includes fifteen SMX-processors and six 64-bit memory controllers. During the program source code compilation, the CUDA compiler introduces the information required for optimum scheduling directly into the GPU instructions. Furthermore, the new SMX-processor makes it possible to execute the double-precision floating-point instructions concurrently with standard instructions.

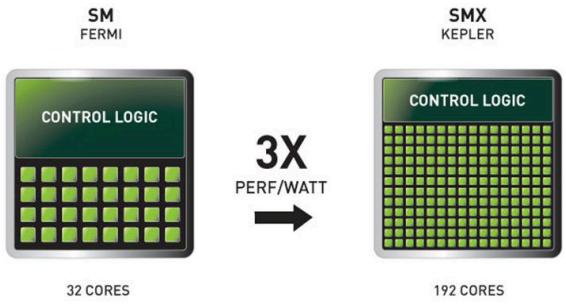


***Fig.2.*** *SMX:192 CUDA cores, 32 Shecial Fanction Units (SFU)*

## 4. DYNAMIC PARALLELIZATION (PARALLELISM, MULTISEQUENCING) OF DATA STREAMS

The new option offered by the Kepler architecture, viz., the so-called dynamic parallelization of data streams, permits one to develop concurrent child processes during runtime of the GPU-program, and also, to organize their synchronization directly within the GPU, without the CPU call (Fig.3).

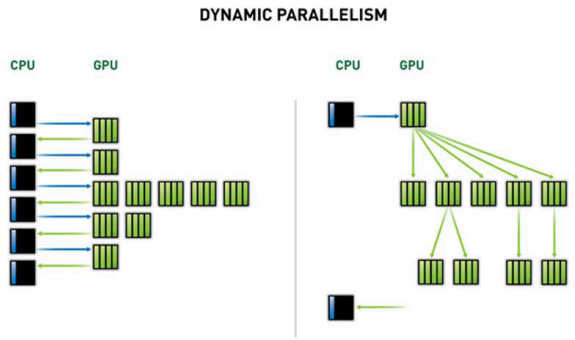


***Fig.3.*** *Static (left side) and Dynamic Parallelism (right side)*

The CUDA dynamic parallelism also provides the GPU realization of hierarchical algorithms, where the structures of subsequent levels of computations and the data for them are determined and formed by the GPU means, without reference to the CPU (Fig.4).

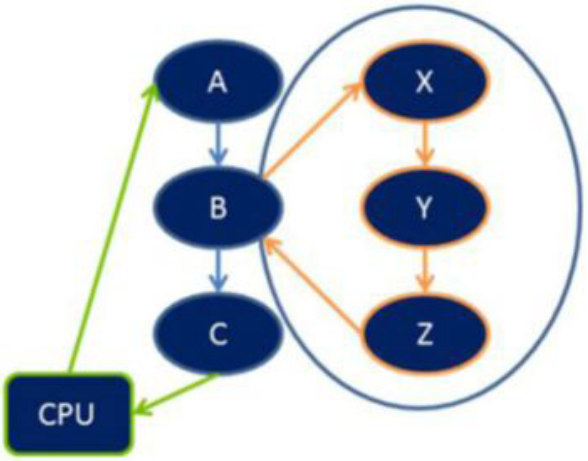


***Fig.4.*** *Static (left side) and Dynamic Parallelism (right side)*

This makes it possible to realize the recursive algorithms, the dynamic structures of cycles and other software constructions within the GPU, thereby enabling, e.g., an easy speed-up of nested parallel cycles. The realization of such algorithmic constructions within the CUDA kernel shortens drastically the traffic due to exclusion of data duplication between the CPU and GPU, and thus speeds up the operation of software constructions using these options. To make use of the advantages of dynamic parallelization of data streams, the CUDA software configuration was extended to include the support means of streams creation and synchronization (Fig.5).

```
__global__ ChildKernel(void* data)
{ //Operate on data } __global__
ParentKernel(void *data)
{ if (threadIdx.x == 0) {
ChildKernel<<<1, 32>>>(data);
cudaThreadSynchronize(); }
```

```
__syncthreads();
//Operate on data
}
// In Host Code ParentKernel<<<8,
32>>>(data);
```

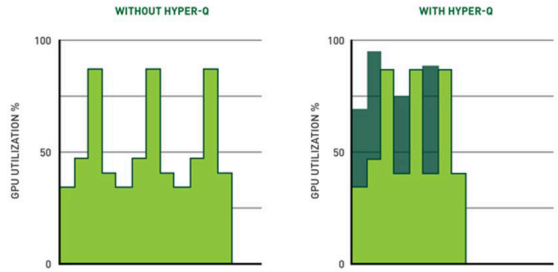


***Fig.5.*** *GPU management for streams control*

## 5. HYPER-Q TECHNOLOGY

A rather complicated problem that prevented the full use of GPU capabilities for scientific computations was a weak optimization of executing parallel computational streams. The Fermi architecture provided simultaneous execution of up to 16 independent streams, but in this case a simple hardware resources multiplexing was ensured within a single request queue. In the case of stream interdependences, the SMD-processors occupied by a waiting stream were simply suspended until the execution of the scheduled stream was completed. The optimization of the GPU loading resulted in a sharp sophistication of the application program and was insufficiently efficient. The Hyper-Q capability has been realized in the Kepler architecture. It is similar to the existing technology for the CPU processors, which is known as Hyper-threading [3]. The physical processor that realizes this technology can keep at once the state of two independent streams. For the operating system, that looks as the presence of two logical processors. Physically, each of the logical processors has its own register set and the interrupt controller (APIC), and the rest processor elements (generally, the address calculation unit and arithmetic units) are shareable. If during the execution of the stream by one of the logical processors the pause arises (as a result of cache miss, the branch prediction error, previous instruction outcome expectation), then the control is transferred to the stream in the other logical processor. Thus, while one process is waiting, for example, for the memory data, the computational resources of the physical processor are used for the running the other process. The realization of this architecture for the INTEL processors has increased the processor throughput by 30...50% at only a 10% increase in the chip area. A similar Hyper-Q technology [4] realized in the GPU GP110 based on the Kepler architecture provides a simultaneous operation of several CPU computation kernels with a single GPU processor. Each CUDA stream has its own GPU apparatus queue. In the device, the stream dependences are optimized, and the waiting on the termination of operation in the one stream no longer leads to blocking the execution of the other stream. And thus, no additional efforts in the development of the program are required for organizing the competitive execution of independent streams. Hyper-Q allows one to increase drastically the GPU loading, and also, to reduce the CPU idle periods. In the case of GPU GP110, this provides the possibility of processing at a time up to 32 hardware-controlled independent streams (communications) between the CPU and GPU Kepler GK 110 (compare only one communication in the Fermi case). Owing to the use of Hyper-Q, the multithread applications, earlier restricted to the capabilities of only one GPU, can now increase the computation speed by a factor of up to 32 without any modifications of the existing programme code [5].

## 6. CONCLUSIONS

The GPU Kepler GK110 architecture much contributes to a high performance computing efficiency. New advantages of the architecture, viz., SMX, Dynamic Parallelism and Hyper-Q, extend the capabilities of hybrid compute-intensive applications, realized through CPU and GPU, and drastically increase their throughput.

# References


1. Whitepaper. NVIDIA's Next Generation. CUDATM Compute Architecture:Kepler TM GK110. 2012 NVIDIA Corporation https://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf

2. Inside 3. David Luebke, Greg Humphreys. *How GPUs Work*. IEEE Computer, February, 2007. IEEE Computer Society, 2007.

3. Pentium 4 Architecture. 2004-11, Hardware Secrets, LLC. All rights reserved. http://www.webcitation.org/61BZRGf1l

4. New line GPU NVIDIA Quadro for architecture NVIDIA Kepler http://www.render.ru/books/show-book.php?book-id=1788

5. CUDA Almanac October 2016 http://www.nvidia.ru/content/EMEAI/images/tesla/almanac/CUDA-almanac-October16.pdf

## НОВЫЕ ВОЗМОЖНОСТИ АРХИТЕКТУРЫ NVIDIA KEPLER И ПЛАТФОРМЫ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ CUDA® И ИХ ИСПОЛЬЗОВАНИЕ ДЛЯ РАЗРАБОТКИ ВЫЧИСЛИТЕЛЬНЫХ НАУЧНЫХ ПРИЛОЖЕНИЙ

***В. А. Дудник, В. И. Кудрявцев, С. А. Ус, М. В. Шестаков***

Описаны дополнительные возможности, предоставляемые новой архитектурой графических процессоров Kepler компании NVIDIA, и применение их для создания высокопроизводительных программ широкого спектра вычислительных научных приложений. Даны рекомендации по их использованию при реализации алгоритмов научно-технических расчётов средствами графических процессоров. Приведено описание новых возможностей платформы параллельных вычислений CUDA®, обеспечивающих набор расширений средств разработки программ для языков Fortran, C и C++, направленных на минимизацию времени разработки приложений и повышение их эффективности.

## НОВІ МОЖЛИВОСТІ АРХІТЕКТУРИ NVIDIA KEPLER І ПЛАТФОРМИ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ CUDA® І ЇХ ВИКОРИСТАННЯ ДЛЯ РОЗРОБКИ ОБЧИСЛЮВАЛЬНИХ НАУКОВИХ ДОДАТКІВ

***В. О. Дуднік, В. І. Кудрявцев, С. О. Ус, М. В. Шестаков***

Описано додаткові можливості, що надаються новою архітектурою графічних процесорів Kepler компанії NVIDIA, і застосування їх для створення високопродуктивних програм широкого спектру обчислювальних наукових застосувань. Надано рекомендації їх використання при реалізації алгоритмів науково-технічних розрахунків засобами графічних процесорів. Приведено опис нових можливостей платформи паралельних обчислень CUDA®, що забезпечують набір розширень засобів розробки програм для мов Fortran, C і C++, направлених на мінімізацію часу розробки додатків і підвищення їх ефективності.