Łukasz STEFANOWICZ, **Remigiusz WIŚNIEWSKI**, Monika WIŚNIEWSKA
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI
ul. Licealna 9, 65-417 Zielona Góra

# Akceleracja obliczeń komputerowych za pomocą układów graficznych z wykorzystaniem technologii CUDA

**Inż. Łukasz STEFANOWICZ**

Student II roku studiów magisterskich. Członek PTI oraz Uczelnianego Koła Naukowego, aktywnie uczestniczący w realizacji pokazów naukowych o zasięgu krajowym oraz międzynarodowym. Zainteresowania obejmują zagadnienia dotyczące programowania oraz projektowania systemów informatycznych, bazodanowych, rozproszonych w środowisku JAVA oraz technologii nVidia CUDA.

*e-mail: lukasz.stefanowicz@yahoo.pl*

**Dr inż. Remigiusz WIŚNIEWSKI**

Absolwent Uniwersytetu Zielonogórskiego, pracę doktorską obronił w 2008 roku. W latach 2000-2001 dwukrotnie odbył przemysłową praktykę studencką w firmie Aldec Inc. w Stanach Zjednoczonych. Aktualnie pracuje jako adiunkt w Uniwersytecie Zielonogórskim. Zainteresowania badawcze obejmują zagadnienia z zakresu kryptologii (zarówno sprzętowej, jak i programowej), oraz metodologii projektowania i implementacji systemów cyfrowych z wykorzystaniem struktur programowalnych FPGA.

*e-mail: R.Wisniewski@iie.uz.zgora.pl*

**Streszczenie**

W artykule zaprezentowano możliwość zastosowania układów graficznych celem przyspieszenia obliczeń komputerowych. Przedstawiono technologię oraz architekturę CUDA firmy nVidia, a także podstawowe rozszerzenia względem standardów języka C. W referacie omówiono autorskie algorytmy testowe oraz metodykę badań, które przeprowadzono w celu określenia skuteczności akceleracji obliczeń komputerowych z wykorzystaniem procesorów graficznych GPU w porównaniu do rozwiązań tradycyjnych, opartych o CPU.

**Słowa kluczowe**: procesor, obliczenia, równoległość, CPU, GPU, CUDA, multimedia, iteracja, wielowątkowość.

## Computing acceleration based on application of the CUDA technology

**Abstract**

The paper deals with application of the graphic processor units (GPUs) to acceleration of computer operations and computations. The traditional computation methods are based on the Central Processor Unit (CPU), which ought to handle all computer operations and tasks. Such a solution is especially not effective in case of distributed systems where some subtasks can be performed in parallel. Many parallel threads can accelerate computing, which results in a shorter execution time. In the paper a new CUDA technology and architecture is shown. The presented idea of CUDA technology bases on application of the GPU processors to computation to achieve better performance in comparison with the traditional methods, where CPUs are used. The GPU processors may perform multi-thread calculation. Therefore, especially in case of tasks where concurrency can be applied, CUDA may highly speed-up the computation process. The effectiveness of CUDA technology was verified experimentally. To perform investigations and experiments, the own test modules were used. The library of benchmarks consists of various algorithms, from simple iteration scripts to video processing methods. The results obtained from calculations performed via CPU and via GPU are compared and discussed.

**Keywords**: processor, compute, parallel, CPU, GPU, CUDA, multimedia, iteration, multithreading.

**Mgr inż. Monika WIŚNIEWSKA**

Ukończyła studia na Wydziale Elektrycznym Uniwersytetu Zielonogórskiego, o specjalności Inżynieria Komputerowa. Obroniła pracę magisterską w 2003 r. W roku 2001 odbyła przemysłową praktykę studencką w firmie Aldec Inc. w Stanach Zjednoczonych. Od 2004 r. jest słuchaczem studiów doktoranckich, specjalność informatyka. Jej zainteresowania naukowe to dekompozycja systemów dyskretnych z wykorzystaniem hipergrafów.

*e-mail: M.Wisniewska@weit.uz.zgora.pl*

## 1. Wstęp

Wraz z biegiem lat można zaobserwować tendencję wzrostu zapotrzebowania na moc obliczeniową. Spowodowane jest to m.in. wzrostem realizmu w projektowanych grach komputerowych, większą złożonością algorytmów przetwarzających, zorientowanych na wyższą jakość obrazów, tekstur, a także chociażby bardziej złożonym systemem interakcji z użytkownikiem (większe możliwości danej aplikacji). Z uwagi na ten fakt pod koniec lat 90. naukowcy z powodzeniem rozpoczęli wykonywanie pierwszych obliczeń numerycznych z użyciem kart graficznych [1, 2]. W ten oto sposób powstawał nurt GPGPU (ang. *General-Purpose Computing on Graphics Processing Units*) zorientowany na wykorzystanie procesorów graficznych do obliczeń nie związanych z grafiką [1, 2, 3, 4, 5]. Tematem zainteresowała się także firma nVidia, która w roku 2007 udostępniła pierwszą wersję zestawu narzędzi dla programistów do obsługi technologii CUDA (ang. *Compute Unified Device Architecture*), a głównym celem było umożliwienie zastosowania kart graficznych jako jednostki obliczeniowej [6, 7]. Niniejszy artykuł ma za zadanie przybliżyć ową technologię, a także poprzez odpowiednio dobrane testy użytkowe wskazać zakres stosowalności.

## 2. Technologia CUDA

Technologia CUDA została zaprojektowana przez firmę nVidia w celu wykorzystania karty graficznej do realizacji obliczeń numerycznych [2]. Główną zaletą przemawiającą za jej stosowalnością jest dużo szybsze wykonywanie obliczeń zmiennoprzecinkowych przez procesor graficzny GPU w porównaniu do procesora klasycznego (CPU). Zastosowanie wielowątkowości sugeruje, że owa technologia sprawdzać się będzie wszędzie tam, gdzie możliwe jest podzielenie zadania na jak największą liczbę wątków [1].

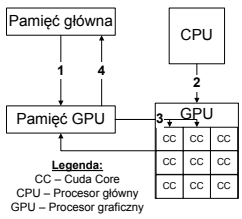
Omawiana architektura posiada wiele zalet. Przede wszystkim wykorzystywany jest język C / C++ (wedle idei "C for CUDA"), dzięki czemu nie ma potrzeby nauki dodatkowego języka programowania. Zaimplementowano operacje bitowe (chociażby przesuwania bitów), w pełni wspierane są liczby całkowite. W przypadku liczb zmiennoprzecinkowych, istnieją pewne odstępstwa od przyjętej normy IEEE 754 względem zaokrąglania liczb [2]. Pamięć współdzielona karty umożliwia swojego rodzaju amortyzację obciążenia związanego z przesyłaniem danych dla poszczególnych wątków. Niestety, w przypadku wysyłania dużej liczby danych do pamięci urządzenia CUDA, przepustowość szyny pomiędzy CPU, a GPU może okazać się wąskim gardłem całej operacji.

Przetwarzanie przez aplikację CUDA można zobrazować poprzez wyszczególnienie czterech etapów (rys.1) [1]:

1) Dane zapisane w pamięci głównej zostają skopiowane do urządzenia CUDA (w tym przypadku karty graficznej z serii GeForce 8 i wyższej).

2) Procesor graficzny GPU zostaje "poinstruowany" przez procesor główny CPU odnośnie realizowanego zadania.
3) Jeden kod funkcji obliczeniowej jest wykonywany na każdym rdzeniu GPU równolegle.
4) Zakończenie obliczeń skutkuje przesłaniem wyników z pamięci urządzenia CUDA do pamięci głównej.

Tak przygotowane dane są następnie analizowane oraz przetwarzane przez procesor CPU.




Rys. 1. Model przetwarzania przez aplikację CUDA
Fig. 1. CUDA application processing model

Zestaw narzędzi programistycznych (CUDA SDK) stanowi swojego rodzaju rozszerzenie języka C o elementy umożliwiające wykonywanie obliczeń przez GPU [3, 4]. Między innymi wprowadzono pojęcie *kernel*, czyli funkcji wykonywanej tylko przez procesor graficzny, będącej typu *void*, wywoływanej przez procesor główny. Argumenty (w tym przypadku dane wejścia / wyjścia) przekazywane są przez referencję [1].

Transfer danych z / do urządzenia CUDA odbywa się z wykorzystaniem funkcji *CudaMemcpy*, natomiast alokacja pamięci na dane wymaga użycia funkcji *CudaMalloc* [1]. Należy zwrócić uwagę na podobieństwo do standardowych funkcji języka C. Wprowadzone zostały także specyficzne typy danych, zorientowane na przetwarzanie równoległe [4].

## 3. Realizacja obliczeń z wykorzystaniem procesora CPU

Istniejący schemat przetwarzania danych zakłada, że procesor główny jest odpowiedzialny za wykonanie wszelakich obliczeń. Jest urządzeniem nadrzędnym, przetwarzając dane wejściowe zgodnie z regułą x, wyniku czego generowane są dane wyjściowe [8]. Rozwój procesorów poprzez ograniczenia związane z maksymalnym taktowaniem, jakie jest uzyskiwane poprzez zastosowanie krzemu wpłynął na usprawnianie samej architektury procesora, a także na integrowanie kilku jednostek wykonawczych (rdzeni) w jednej konstrukcji. Obecnie spotykane są procesory dwu, cztero, sześcio oraz ośmiordzeniowe. Konstrukcje firmy AMD cechują się tym, że dany rdzeń przetwarzać może tylko jeden wątek, natomiast procesory firmy Intel - dwa wątki. Należy także zwrócić uwagę, iż CPU nie było projektowane do wykonywania obliczeń, lecz do przetwarzania danych, wskutek czego nie jest jednostką specjalizującą się w operacjach numerycznych.

Integracja większej liczby rdzeni w pojedynczym procesorze wpływa na większe możliwości zrównoleglenia danego algorytmu, dzięki czemu dane zadanie może zostać zrealizowane szybciej. Aktualnie odchodzi się od produkcji CPU zawierających jeden rdzeń na korzyść jednostek wielordzeniowych.

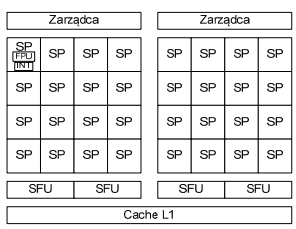
Sekwencyjność wykonywanych instrukcji wiąże się z dużymi ograniczeniami procesora CPU. Jednostka ta jest bardzo mało efektywna w przypadku zaawansowanych obliczeń matematycznych [5]. Dlatego też dość często stosowanym rozwiązaniem są klastry obliczeniowe. Niewątpliwie zaletą takiego podejścia jest akceleracja obliczeń, poprzez rozłożenie jednego zadania na poszczególne elementy systemu. O ile wpłynie to pozytywnie na czas wykonania problemu numerycznego, o tyle koszt całościowy związany z nakładami finansowymi staje się na tyle duży, że nie jest to rozwiązanie osiągalne dla każdego.

Wykorzystanie technologii CUDA umożliwia znaczne odciążenie procesora głównego, poprzez przekazanie obliczeń dla jednostki specjalizującej się w obliczeniach numerycznych (GPU).

## 4. Realizacja obliczeń z wykorzystaniem procesora GPU

O ile procesor główny CPU jest zorientowany na przetwarzanie sekwencyjne, czyli wykonywanie krok po kroku określonych ciągów instrukcji, o tyle procesor graficzny projektowano z myślą równoległego przetwarzania. Jest to wyspecjalizowana jednostka, zdolna do efektywnego przetwarzania wartości liczbowych opisujących dany obraz. Technologia CUDA ujednoliciła możliwości przetwarzania zorientowane na obliczenia numeryczne.

Warto zwrócić uwagę na różnice w budowie - GPU zbudowane jest głównie z multiprocesorów strumieniowych oraz pamięci cache poziomu 1 oraz 2. Zadanie przydziela globalny nadzorca. Bazując na architekturze Fermi, każdy multiprocesor zbudowany jest z 32 procesorów strumieniowych, potocznie nazywanych "rdzeniami CUDA". Każdy taki rdzeń poglądowo oraz w wielkim przybliżeniu można zobrazować jako jeden rdzeń jednostki CPU. Procesory strumieniowe podzielono na dwie grupy po 16, umiejscowiono lokalnego zarządcę oraz dwie jednostki SFU wspierające procesory strumieniowe w bardziej zaawansowanych obliczeniach. Pamięć cache poziomu 1 jest współdzielona przez każdą taką grupę. Rdzeń CUDA zawiera jednostkę zmiennoprzecinkową oraz całkowitą i specjalizuje się w prostych operacjach na liczbach. W przypadku zaistnienia potrzeby wykonania bardziej zaawansowanych kalkulacji pojedyncze procesory strumieniowe wspierane są w obliczeniach przez jednostki SFU [1].




Rys. 2. Budowa Multiprocesora Strumieniowego (SM)
Fig. 2. Streaming multiprocessor (SM) structure

Technologia CUDA zakłada wykorzystanie każdego procesora strumieniowego do wykonania obliczeń. Z uwagi na budowę uwidaczniającą sporą liczbę "rdzeni CUDA" zorientowanie na wielowątkowe przetwarzanie danych jest jak najbardziej oczywiste. Przetwarzanie danych przez GPU opiera się o wątek obliczeniowy, nazwany *kernelem*. Identyczny wątek uruchamiany jest na każdym rdzeniu CUDA, celem uzyskania wyniku części [4]. Jedno złożone zadanie numeryczne rozdziela się pomiędzy poszczególne jednostki, o liczbie znacznie przewyższającej rdzenie w zwykłym procesorze CPU. Różnica ta skutkować ma znacznie krótszym czasem uzyskania wyniku.

Architektura CUDA wymaga od projektanta aplikacji wykonywania specyficznych kroków, takich jak przesył danych do pamięci karty graficznej przed wykonaniem obliczeń, a także przetransferowania wyników z pamięci graficznej do pamięci głównej, aby móc zinterpretować wyniki. Od funkcji obliczeniowej wymaga się, aby była jak najmniejsza oraz zorientowana na jak największą liczbę obliczeń. Wykonywanie sekwencyjne (pętle), czy instrukcje warunkowe powinny być używane w ostateczności, ponieważ spowalniają one dodatkowo proces obliczeniowy [5].

## 5. Badania i eksperymenty

Faza eksperymentalna zakładała wykonanie badań mających na celu wyznaczenie obszaru stosowalności technologii CUDA, poprzez pomiar czasu wykonywania poszczególnych algorytmów zarówno tradycyjnych (wykorzystanie CPU), jak i zorientowanych na przetwarzanie masowo równolegle (GPU). Wyszczególnione zostały następujące obszary zainteresowań:

1) Operacje iteracyjne, brak obliczeń.
2) Operacje równoległe, z uwzględnieniem obliczeń na liczbach zmiennoprzecinkowych.

3) Operacje masowo równoległe, uwzględniające obliczenia numeryczne.

Obszar pierwszy zakładał wykorzystanie CPU oraz GPU w prostych operacjach iteracyjnych, jak dodawanie wektorów. Zakładana tablica zawierała odpowiednio dużą ilość elementów, aby móc wyznaczyć różnice czasowe związane z realizacją zadania.

Obszar drugi zakładał równoległą realizację zadań punktu pierwszego poprzez podział operacji na poszczególne rdzenie. W przypadku przypisywania elementom tablicy wartości, dany element wyznaczany był przez zdefiniowaną funkcję matematyczną.

Obszar trzeci zakładał pomiar czasu związany z realizacją wyspecjalizowanych zadań, jak:

a) przetwarzanie pliku multimedialnego zawierającego dźwięk oraz obraz (avi) algorytmem H.264 celem uzyskania wynikowego pliku (mp4),
b) korekcja pliku multimedialnego zawierającego dźwięk oraz obraz, poprzez operacje w czasie rzeczywistym mające na celu korekcję wyświetlanych danych (redukcję szumu, poprawę kontrastu, korekcję palety kolorów),

Mierzony czas tyczył się wykonywania całej operacji obliczeniowej przez GPU, składającej się z:

1) Wysłania danych do pamięci urządzenia CUDA.
2) Wykonania obliczeń przez urządzenia CUDA.
3) Pobrania danych do pamięci głównej.

Wszelkie testy realizowane były na komputerze wyposażonym w procesor główny AMD Phenom II X4 B55 3.4 GHz, 4 GB pamięci RAM pracującej przy częstotliwości 1600 MHz o opóźnieniach 7-7-7-21 1T. Zestaw wyposażony był w kartę graficzną EVGA GeForce 8800 GTS 512 MB (G92) [6].

Obszar pierwszy:

1) iteracyjne dodawanie wektorów

| Liczba elementów | Czas CPU [ms] | Czas GPU [ms] |
|---|---|---|
| 10 mln | 0,005 | 0,02 |
| 20 mln | 0,01 | 0,04 |
| 100 mln | 0,05 | 0,09 |

Test wykazuje przewagę rozwiązania klasycznego w przypadku zadania iteracyjnego. Procesor główny CPU lepiej radzi sobie z typowymi zadaniami, nie wymagającymi znacznego nakładu obliczeniowego.

Obszar drugi:

2) równoległe dodawanie wektorów

| Liczba elementów | Czas CPU [ms] | Czas GPU [ms] | Ilość wątków |
|---|---|---|---|
| 10 mln | 0,0006 | 0,002 | 10 |
| 10 mln | 0,0005 | 0,0009 | 20 |
| 20 mln | 0,0007 | 0,003 | 10 |
| 20 mln | 0,0006 | 0,001 | 20 |
| 100 mln | 0,005 | 0,008 | 10 |
| 100 mln | 0,005 | 0,003 | 20 |

W dalszym ciągu można zauważyć przewagę rozwiązania klasycznego. Jedynie w przypadku sporej ilości danych oraz zwiększeniu liczby wątków, GPU zdaje się lekko wyprzedzać CPU. Bazując na tym wniosku można stwierdzić, iż dalsze zwiększanie ilości wątków skutkować będzie dalszym zyskiem.

Obszar trzeci:

3a) przetwarzanie pliku video algorytmem H.264 (końcowa rozdzielczość: 320x240)

| Rozdzielczość początkowa | Długość [s] | Czas CPU [s] | Czas GPU [s] | Kodek |
|---|---|---|---|---|
| 640x480 | 605 | 114 | 62 | DIVX |
| 592x312 | 6180 | 1084 | 459 | XVID |
| 1280x544 | 6420 | 1634 | 1041 | Matroska MPEG4 |

Pierwsze testy wykazały znaczną przewagę GPU nad CPU. Daje się zauważyć, że wszędzie tam, gdzie zadania mogą być realizowane w sposób wielowątkowy, a także narzut obliczeń jest znaczny, tradycyjne rozwiązanie zdaje się być niewystarczające.

3b) korekcja pliku video w czasie rzeczywistym

| Rozdzielczość | Długość filmu [s] | Liczba klatek / s (CPU) | Liczba klatek / s (GPU) | Szybkość klatek |
|---|---|---|---|---|
| 592x312 | 6180 | 9 | 25 | 25 |
| 640x272 | 6240 | 2,8 | 12 | 25 |
| 720x304 | 5220 | 4,9 | 19 | 25 |

W przypadku przetwarzania plików multimedialnych w czasie rzeczywistym zysk z zastosowania technologii CUDA jest większy, niż przy przetwarzaniu plików z użyciem algorytmu H.264. Mowa tutaj o wydajności od 2.5 do 4.5 wyższej. Warto zauważyć, iż w tym przypadku GPU jest zawsze sporo wydajniejsze od CPU.

## 6. Wnioski

Niniejszy artykuł miał na celu przybliżyć technologię CUDA firmy nVidia oraz wprowadzić w tematykę przetwarzania równoległego. Przedstawiono pokrótce architekturę oraz sposób przetwarzania danych przez procesory CPU oraz GPU. Omówiono także model aplikacji CUDA. Zaplanowane eksperymenty miały na celu wykazać realną wydajność algorytmów bazujących na rozwiązaniu tradycyjnym, czyli CPU, oraz w pełni równoległym GPU. Okazuje się, że problemy iteracyjne, wykonywane są szybciej przez procesor CPU. Zadania takie praktycznie nie wykorzystują mocy procesora graficznego, a potrzeba przesyłania danych do jednostki GPU wydłuża całkowity czas realizacji operacji. W przypadku zadań wielowątkowych procesor graficzny wykazał znaczną przewagę nad procesorem CPU, co wynikało z pełnej możliwości wykorzystania jego architektury. Operacje wielowątkowe umożliwiają akcelerację zadania, niezależnie od złożoności obliczeniowej. Oczywiście im więcej składowych do policzenia ma dany kernel, tym bardziej efektywnie względem CPU będzie to realizowane.

Podsumowując należy stwierdzić, że opisywana technologia zastosowania procesora graficznego jest szczególnie efektywna w przypadku zadań wielowątkowych. Biorąc pod uwagę koszt pojedynczej karty graficznej, pokazane rozwiązanie wydaje się znacznie bardziej opłacalna niż tworzenie drogich systemów rozproszonych. Technologia CUDA rozwija się na tyle dynamicznie, że wiele producentów oprogramowania stosuje ją z powodzeniem w celu redukcji czasu oczekiwania na rezultat końcowy [7]. Wydaje się, że w przyszłości większość oprogramowania korzystać będzie z dobrodziejstw przetwarzania równoległego z użyciem procesora graficznego.

## 7. Literatura

[1] Kirk D. B., Hwu W. W.: Programming Massively Parallel Processors A Hands-on Approach, MK, Burlington, USA, 2010.
[2] Sanders J., Kandrot E.: CUDA by Example, AW, Ann Arbor, USA, 2010.
[3] http://developer.download.nvidia.com/compute/cuda/3_2_prod/toolkit/docs/ CUDA_C_Best_Practices_Guide.pdf
[4] http://developer.download.nvidia.com/compute/cuda/3_2_prod/toolkit/docs/ CUDA_C_Programming_Guide.pdf
[5] Fulmański P., Popa A., Witczak D.: Highly Parallel and Arithmetic Intensified Computation With Graphics Card
[6] http://www.nvidia.pl/page/geforce8.html
[7] http://www.benchmark.pl/testy_i_recenzje/nVidia_CUDA_i_ATI_Stream._Wykorzystac_moc_GPU-1942/strona/5364.html
[8] Biernat J.: Architektura komputerów, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2005.

*otrzymano / received: 13.05.2011*
*przyjęto do druku / accepted: 04.07.2011* *artykuł recenzowany*