Maciej WIELGOSZ, Ernest JAMRO, Kazimierz WIATR
AKADEMIA GÓRNICZO-HUTNICZA, ACK - CYFRONET
AKADEMIA GÓRNICZO-HUTNICZA, KATEDRA ELEKTRONIKI

Akceleracja obliczeń zmiennoprzecinkowych na platformie RASC

Mgr inż. Maciej WIELGOSZ

Ukończył studia na AGH (2005), wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki na kierunku Elektronika i Telekomunikacja. Obecnie jest doktorantem w Katedrze Elektroniki AGH i bierze czynny udział w pracach badawczych realizowanych w zespole rekonfigurowalnych systemów obliczeniowych. Jego zainteresowania naukowe dotyczą sprzętowej akceleracji obliczeń, kompresji obrazu i sieci neuronowych.

e-mail: wielgosz@agh.edu.pl

Dr inż. Ernest JAMRO

Ukończył studia na AGH na kierunku Elektronika oraz na University of Huddersfield (UK) na kierunku Elektronika i Telekomunikacja. Obronił pracę doktorską w 2001 roku na AGH na wydziale Elektrotechniki, Automatyki, Informatyki i Elektroniki. Aktualnie jest adiunktem w Katedrze Elektroniki na AGH. Jego zainteresowania naukowe to sprzętowa akceleracja obliczeń, niskopoziomowe przetwarzanie obrazów, sieci neuronowe.

e-mail: jamro@agh.edu.pl

Prof. dr hab. inż. Kazimierz WIATR

Studia AGH Kraków (1980), dr nauk technicznych (1987), dr habilitowany (1999) i profesor (2002). Profesor zwyczajny na Akademii Górniczo-Hutniczej oraz Dyrektor Akademickiego Centrum Komputerowego Cyfronet AGH. Prowadzone prace badawcze dotyczą komputerowego sterowania procesami, systemów wizyjnych, systemów wieloprocesorowych, układów programowalnych, rekonfigurowalnych systemów obliczeniowych i sprzętowych metod akceleracji obliczeń.

e-mail: wiatr@agh.edu.pl

Streszczenie

W artykule zostały zaprezentowane wyniki testów przeprowadzonych w celu określenia maksymalnej szybkości wykonywania operacji zmiennoprzecinkowych na platformie rekonfigurowanej RASC. Zaimplementowano różne dostępne tryby konfiguracji jednostki Host oraz RASC w celu wyłonienia najbardziej efektywnego pod względem wydajności trybu pracy jednostki obliczeniowej. Uzyskane wyniki pomiarów ujawniały, że kombinacja Direct I/O oraz DMA zapewnia najwyższą przepustowość pomiędzy węzłami Host i RASC. Niemniej jednak dla niektórych aplikacji tryb multi-buffering może okazać się bardziej odpowiedni, ze względu na możliwość jednoczesnego przesyłania danych i wykonywania operacji. Funkcja exp() w standardzie zmiennoprzecinkowym o podwójnej precyzji została wykorzystana jako przykładowa aplikacja, która pozwoliła oszacowanie możliwej do uzyskania akceleracji obliczeń na platformie RASC.

Słowa kluczowe: Akceleracja sprzętowa, Komputery dużej mocy (HPC), FPGA, obliczenia zmiennoprzecinkowe, funkcja exp()

Accelerating calculations on the RASC platform

Abstract

This paper presents results of the test performed to determine high speed calculations capabilities of the SGI RASC platform. Different data transfer modes and memory management approaches were examined to choose the most effective combination of the Host and RASC memory adjustments. That work may be regarded as a case study of the contemporary FPGA - based accelerator, which however can characterize whole branch of the devices. The paper is focused strongly on the floating point calculations' potential of the FPGA accelerator. The RASC algorithm execution procedure, from the processor's perspective, is composed of several functions which reserve resources, queue commands and perform other preparation steps. It is noteworthy (Fig.3) that the time consumed by the functions remains roughly the same, independent of the algorithm being executed. The resource reservation procedure, once conducted, allows many executions of the algorithm -that amounts to huge time savings, since the procedure takes approximately 7.5 ms, which is roughly 99 % of overall execution time of the algorithm. Rasclib algorithm commit and rasclib algorithm wait calls are considered to be the key (Fig.3) part of the RASC software execution routine. The first one activates the FPGA algorithm, the second one waits for the completion flag. The period between these two commands is the transfer and algorithm execution time. All curves (Fig.4) reflect overall processing time of the same amount of data, but differ in size of the single data chunk, which varies from 1024x64 bit = 8 kB to 1048576x64 bit = 8 MB. It has been observed that for the bigger chunk much better results are achieved in terms of effective execution time. However above 1 MB, a decrease of effective execution time seems to indicate saturation, therefore sending data in bigger portions may not improve the performance of the system so much. The most effective execution time of single exp() function for SRAM buffering mode is 12 ns, so 9,5 ns is transport overhead due to bus delays. Theoretical calculation time of single exp()function (data transfer is not taken into account) is 2,5 ns because two exp() are implemented on the RASC and clocked at 200 Mhz. Obtained results of measurements reviled that Direct I/O mode together with DMA transfer provides the highest data throughput between the Host and RASC slice. Nevertheless, for some application multi-buffering may appear to be more suitable in terms of concurrent data transfer capabilities and FPGA algorithm execution. As a hardware acceleration example, the exponential function is taken which allows estimating maximum achievable data processing speed.

Keywords: HPRC (High Performance Reconfigurable Computing), FPGA, elementary functions, exponential function

1. Wstęp

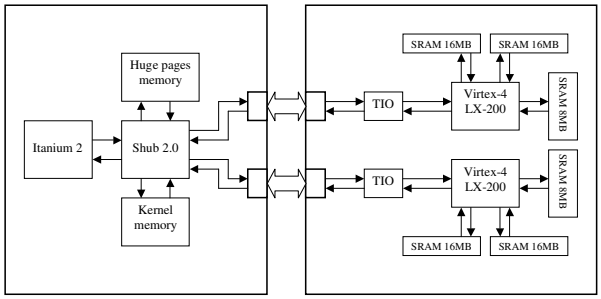
Od wielu lat prowadzone są badania nad wykorzystaniem układów FPGA w systemach HPC jako pełnowartościowych modułów obliczeniowych zaszytych głęboko w systemie komputerowym. Warto wspomnieć o kilku ważnych ograniczeniach, które do tej pory uniemożliwiały szersze wykorzystanie układów FPGA w komputach dużej mocy. Po pierwsze obliczenia HPC wykonywane są zazwyczaj w standardzie liczb zmiennoprzecinkowych, które wymagają dość znaczących zasobów logicznych a pojemność logiczna układów FPGA była relatywnie mała. Bardzo szybki wzrost pojemności logicznej nowoczesnych układów programowalnych rozwiązuje ten problem. Kolejnym wyzwaniem jest silna konkurencji ze strony kart graficznych, które coraz częściej znajdują zastosowanie jako akceleratory [2] szczególnie operacji zmiennoprzecinkowych pojedynczej precyzji. Niemniej jednak, wielu znaczących producentów akceleratorów sprzętowych (SGI, SRC, DRC, Xtrem-Data) [1] skłania się w kierunku wykorzystania układów FPGA w system HPC.

Autorzy niniejszego artykułu od kilku lat prowadzą badania nad przyspieszaniem algorytmów kwantowo-chemicznych. Ostatnio do ich dyspozycji została oddana nowoczesna platforma obliczeniowa Altix 4700 i układem RASC [1]. W wyniku prowadzonych badań uzyskano serie interesujących wyników, które umożliwiły oszacowanie prędkości obliczeniowej modułu RASC oraz potencjału układów programowalnych FPGA w systemach HPC. Jako projekt testowy wykorzystano funkcję exp() podwójnej precyzji. Zajmuje ona tylko około 5 % całego układu programowalnego Xilinx Virtex-4 [5]. Umieszczenie jednak dwóch równoległych modułów obliczających exp() absorbuje w pełni dostępną szerokość magistrali danych (128 bit) pomiędzy układem FPGA a pamięcią zewnętrzną uniemożliwiając uzyskanie większego stopnia zrównoleglenia a przez to większej akceleracji obliczeń. Warto wspomnieć, że wybór funkcji exp() nie jest przypadkowy, jest ona

bardzo popularna w obliczeniach kwantowo chemicznych oraz fizycznych, dlatego jej akceleracja stanowi spory przyczynek do przyspieszenia całej aplikacji [3]. Ponadto akceleracja obliczeń dla funkcja *exp()* jest ograniczona raczej przez transfer danych do/z układu FPGA a nie przez dostępne zasoby układu FPGA. Szybkość wykonywania obliczeń *exp()* jest w przybliżeniu taka sama jak gdyby dane były kopiowane z jednego bloku pamięci do drugiego z udziałem układu FPGA.

2. Superkomputer Altix 4700 z modułem RASC

SGI Altix 4700 jest wieloprocesorowym superkomputerem DSM (distributed shared memory), który może zostać rozbudowany do pojemności 512 procesorów Itanium oraz 6 TB współdzielonej pamięci operacyjnej. Do systemu również dołączane są rekonfigurowane moduły RASC, które komunikują się z pozostałą częścią jednostek obliczeniowych za pośrednictwem interfejsu NUMA-link (non-uniform memory access link) na równych prawach z innymi procesorami.

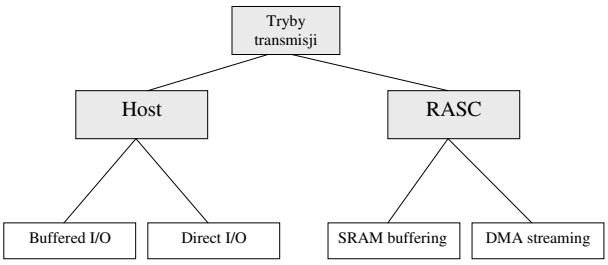


Rys. 1 Struktura platformy obliczeniowej z układem RASC
Fig. 1 RASC based accelerator block diagram

Jednostka RASC składa się z dwóch układów Xilinx Virtex-4 LX200. Na jeden układ FPGA przypada jeden moduł TIO (układ ASIC zapewniający komunikacje z resztą systemu poprzez magistralę NUMA-link) oraz pięciu synchronicznych pamięci SRAM, zgrupowanych logicznie w trzy struktury.

Uruchomienie akceleratora RASC następuje z procesora (Host) poprzez odpowiednio zdefiniowane funkcje systemowe. Procesor Host nadzoruje również proces dostarczenia danych do węzła obliczeniowego RASC oraz odebrania wyników.

W omawianej w niniejszym artykule aplikacji w jednym układzie FPGA zostały zaimplementowane dwie równoległe funkcje exp(), każda z nich przetwarza słowo 64 bitowe, co wymaga dostarczenia i odebrania sumarycznie 128 bitów co takt zegara. Moduł RASC pracuje z zegarem 200 MHz. SGI Altix 4700 dysponuje czterema trybami transmisji oraz zarządzania danych. Dwa z nich konfigurowane są po stronie procesora Itanium (Host) a pozostałe dwa po stronie układu FPGA (RASC).



Rys. 2 Dostępne tryby transmisji dla platformy RASC
Fig. 2 Different data transfer modes available on Altix 4700 machine with the RASC slice

Warto zaznaczyć, że jednej platformie RASC dostępne są dwa układy FPGA, ale producent nie zapewnił komunikacji pomiędzy nimi bezpośrednio na płycie. Istnieje możliwość przesłania danych pomiędzy układami, ale wymaga to skorzystania z zewnętrznego Huba komunikacyjnego.

3. Wpływ trybu transmisji na uzyskany wynik akceleracji

Sram buffering

Ten tryb transmisji danych wymaga alokacji danych w jednej z trzech pamięci zewnętrznych SRAM modułu RASC [Rys.1]. Po przetransmitowaniu danych, FPGA rozpoczyna ich przetwarzanie i jednocześnie wynik zapisywany jest do osobnego bloku pamięci. Warto zwrócić uwagę, że operacji przesłania danych, wykonania obliczeń i odczytania danych są wykonywanie sekwencyjnie w przeciwieństwie do trybu m*ulti-buffering*, w którym wszystkie one są realizowane jednocześnie.

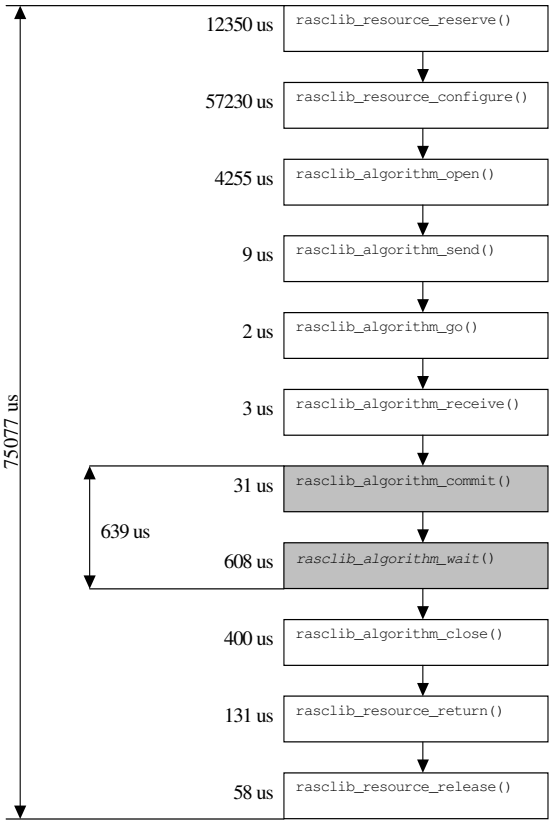
Z punktu widzenia procesora wykonanie algorytmu realizowanego na platformie RASC wiąże się z uruchomieniem kilku funkcji z biblioteki *rasc_lib* (m.in. rezerwującej zasoby systemu, kolejkującej komendy). Autorzy przeprowadzili dokładne pomiary czasu wykonania poszczególnych funkcji dla przykładowego wektora danych obliczeniowych. Parametrem decydującym o uzyskiwanej akceleracji, jest efektywny czas wykonania operacji. Czynnikami w znacznym stopniu determinującymi ten czas jest właśnie szereg operacji inicjujących pracę akceleratora, producenci często unikają podawania tych parametrów. SGI [1] nie jest tym przypadku wyjątkiem.

Na podstawie uzyskanych wyników stwierdzono, że czas wykonania funkcji rezerwujących zasoby akceleratora RASC pozostaję prawie niezmienny i wynosi około 7,5 ms, główną jego składową jest czas programowania układu FPGA. Co stanowi około 99 % czasu wykonania algorytmu. Dlatego bardzo ważne jest, aby procedurę inicjalizacji zasobów przeprowadzać jak najrzadziej.

Funkcję *rasclib_algorithm_commit()* oraz *rasclib_algoritm_wait()* [Rys. 3] można uznać za kluczowe. Pierwsza z nich uruchamia wykonanie algorytmu na platformie RASC, druga natomiast oczekuje na flagę zakończenia wykonania obliczeń. Czas pomiędzy uruchomieniem tych dwóch funkcji jest w pełni zależny do zaimplementowanego algorytmu oraz ilości przetwarzanych danych.

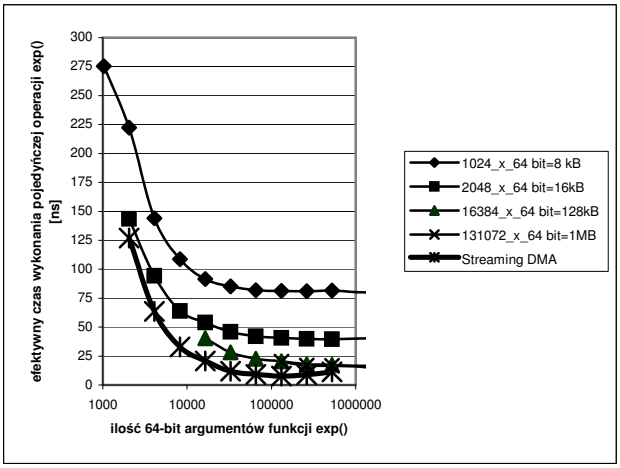
Warto zwrócić uwagę, że równocześnie z obliczeniami prowadzonymi na platformie RASC, można uruchomić odrębną część algorytmu na procesorze Host, gdyż pomiędzy wykonaniem funkcji *rasclib_algorithm_commit()* oraz *rasclib_algoritm_wait()* węzeł sterujący nie jest obciążony. Stosując takie podejście można uznać kolejny sposób zrównoleglenia realizacji algorytmu. Istotne jednak jest, aby czas wykonania części softwarowej nie przekroczył czasu realizacji obliczeń w akceleratorze FPGA. Takie zaprojektowanie algorytmu wymaga znajomości specyfiki platformy pod kątem oczekiwanych czasów realizacji operacji.

Na podstawie dokonanych pomiarów stwierdzono, że przesyłanie danych w większych pakietach prowadzi do lepszych rezultatów, jeśli chodzi o efektywny czas wykonania operacji. Jednak dla pakietów o rozmiarach większych niż 1 MB nie następuje dalsze zwiększanie wydajności transmisji, dochodzi do nasycenia. Najkrótszy czas wykonania funkcji *exp()* w przeliczeniu na jedną operację, jaki udało się uzyskać w trybie *sram buffering* wynosi 12 ns. Możemy zauważyć, że narzut związany z transferem w tym przypadku równa się 9,5 ns, gdyż teoretyczny czas wykonania pojedynczej operacji *exp()* wynosi 2,5 ns (dwie operacje exp() wykonywane z częstotliwości 200 MHz).



Rys. 3 Czasy wykonania poszczególnych funkcji odpowiedzialnych za sterowanie pracą akceleratora RASC
Fig. 3 Time estimations of RASC functions execution

Wszystkie prezentowane przebiegi [Rys.4] przedstawiają całkowity czas wykonania operacji *exp()* dla pojedynczej danej 64 bitowej. Za każdym razem przesyłana jest taka sama ilość danych, ale w pakietach o różnych rozmiarach, wahających się od 1024×64 bit = 8kB do 1310726×64 bit = 1 MB.



Rys. 4 Efektywny czas wykonania pojedynczej operacji exp()
Fig. 4 Effective execution time of single exp() function

DMA streaming

Zastosowanie trybu *DMA streaming* oznacza rezygnacje z alokowania danych w pamięci SRAM znajdującej się na płycie RASC, pamięci, która jest na zewnątrz układu FPGA. Takie podejście pozwala zaoszczędzić sporo czasu, który jest potrzebny na transfer danych pomiędzy układem FPGA a pamięcią. Napływające dane są czytane bezpośrednio ze strumienia wejściowego DMA i zapisywane do drugiego niezależnego strumienia, to pozwala skrócić efektywny czas wykonania jednej operacji *exp()* do 7,8 ns w porównaniu, z 12ns dla trybu *sram buffering*.

4. Dobór trybu zarządzania pamięcią procesora (Host)

System operacyjny dąży do jak najlepszego wykorzystania ograniczonej ilości dostępnych TLB (Translation Lookaside Buffer). Optymalizacja wykorzystania tych zasobów ma tym większe znaczenie im większą pamięcią dysponuje system. Strona pamięć typu *hugepages* jest wielokrotnie większa niż zwykła strona. Alokowanie danych po stronie Hosta w tym właśnie obszarze może prowadzić do zwiększenia wydajności sięgającego nawet 30%. Po stronie jednostki Host dostępne są dwa tryby konfiguracji alokacji danych wysyłanych do akceleratora sprzętowego: Buffered i Direct.

Tryb buffered I/O

Wybór tego trybu prowadzi do umieszczenia danych w przestrzeni pamięci użytkownika. Niestety transfer danych w trybie buffered I/O wiąże się z dodatkowym ich kopiowaniem do pamięci nadzorowanej przez jądro systemu, co zmniejsza przepustowość sytemu.

Tryb Direct I/O

Dane przesyłane do akceleratora RASC zostają wstępnie umieszczone w pamięci *hugepages*. Transfer danych z pamięci typu *hugepages* zajmuje znacznie mniej czasu niż transmisja w trybie buffered I/O.

5. Wnioski

Zagadnienie transferu danych w systemach zawierających akceleratory sprzętowe jest szeroko dyskutowane w literaturze [4] i stanowi podstawowe kryterium oceny danego rozwiązania. W nowoczesnych systemach moc obliczeniowa węzłów składowych jest duża, natomiast wydajność systemu jako całości może być rzetelnie oceniona tylko po uwzględnieniu wszystkich wąskich gardeł w transmisji danych.

Deklarowana przez SGI teoretyczna przepustowość 3,2 GB/s nie została osiągnięta. Na podstawie przeprowadzonych badań konfiguracja Direct I/O oraz DMA streaming pozwala na uzyskanie najkrótszego czasu wykonania pojedynczej operacji *exp()*. Ta sama operacja wykonana na procesorze Itanium zajmuje około 11 razy więcej czasu.

Należy również zauważyć, że wyniki akceleracji będą znacznie lepsze w miarę zwiększania ilości stopni przetwarzania (stopnia skomplikowania algorytmu) zaimplementowanego w układzie FPGA.

6. Literatura

[1] Silicon Graphics, Inc. Reconfigurable Application-Specific Computing User's Guide, Ver. 005, January 2007, SGI
[2] M. Giles, *GPU's - the next big advance in HPC?*, Oxford University, Reconfigurable Supercomputing Conference (MRSC), 1-3 April 2008, Belfast, Northern Ireland
[3] M.Wielgosz, E. Jamro, K. Wiatr, *Highly Efficient Structure of 64-Bit Exponential Function Implemented in FPGAs,* ARC 2008, Lecture notes in Springer-Verlag, London LNCS 4943, pp. 274 – 279
[4] Underwood, K. D., Hemmert, K. S., and Ulmer, C.: Architectures and APIs: assessing requirements for delivering FPGA performance to applications. Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (Tampa, Florida, November 11 - 17, 2006).
[5] Xilinx Virtex-4 Family Overview http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf

Artykuł recenzowany