AUTOMATYKA • 2009 • Tom 13 • Zeszyt 3

Maciej Wielgosz*, Ernest Jamro*, Kazimierz Wiatr*

Implementacja w układach FPGA modułu obliczającego funkcję jednoelektronową

1. Wprowadzenie

Od wielu lat prowadzone są badania nad wykorzystaniem układów programowalnych FPGA (*Field Programmable Gate Array*) w HPC (*High Performance Computing*) jako akceleratorów sprzętowych. Obliczenia HPC wykonywane są zazwyczaj w standardzie liczb zmiennoprzecinkowych podwójnej precyzji, wymagających dość znaczących zasobów logicznych, których ilość była relatywnie mała. W ostatnich latach jednak obserwuje się znaczący wzrost zasobów układów logiki rekonfigurowalnej, dlatego też wielu producentów akceleratorów sprzętowych (SGI, SRC, DRC, Xtrem-Data) [1] skłania się w kierunku wykorzystania układów FPGA w systemach HPC.

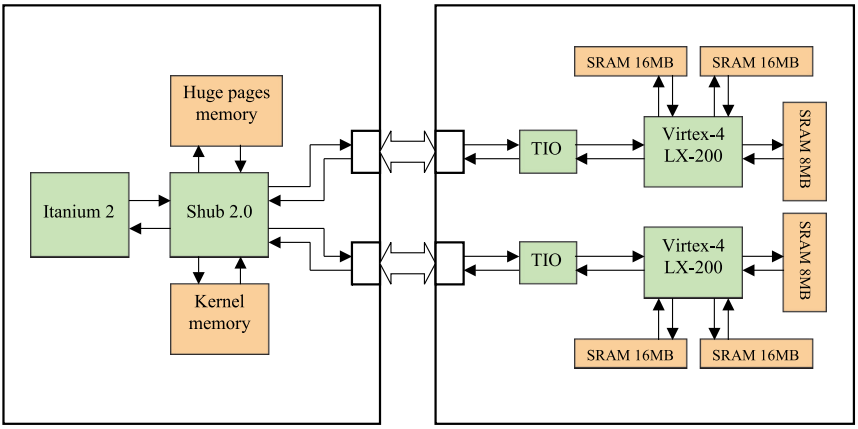
Autorzy niniejszego artykułu od kilku lat prowadzą badania nad przyspieszaniem algorytmów kwantowo-chemicznych. W wyniku licznych konsultacji wybrano część eksponencjalną funkcji orbitalnej (równanie (3)) do akceleracji sprzętowej. Operacja ta jest bardzo często spotykana w obliczeniach kwantowo-chemicznych i może być wykorzystania również w innych systemach obliczeniowych, nad którym pracują autorzy niniejszej pracy (np. metoda kwantowego Monte Carlo [3, 4]).

Autorom nie jest znana podobna sprzętowa akceleracja. Należy również nadmienić, że wyniki prezentowane w niniejszej pracy stanowią pierwszy etap sprzętowej akceleracji całej funkcji orbitalnej. Ostatecznym celem realizowanego projektu jest pełna implementacja modułu realizującego operację całkowania potencjału korelacyjno-wymiennego. Procedura ta jest wieloetapowa i zostanie pokrótce omówiona na łamach niniejszego artykułu.

2. Platforma obliczeniowa

SGI Altix 4700 jest wieloprocesorowym superkomputerem DSM (*Distributed Shared Memory*), który może zostać rozbudowany do 512 procesorów Itanium oraz 6 TB współdzielonej pamięci operacyjnej. Do systemu dołączane są również rekonfigurowane moduły RASC (*Reconfigurable Application Specific Computing*), które komunikują się z pozostałą częścią jednostek obliczeniowych za pośrednictwem interfejsu NUMA-link (*Non-Uniform Memory Access Link*) na równych prawach z innymi procesorami.

* Katedra Elektroniki, Akademia Górniczo-Hutnicza w Krakowie, ACK-CYFRONET, Kraków



Rys. 1. Struktura platformy obliczeniowej z układem RASC

Jednostka RASC [1] składa się z dwóch układów Xilinx Virtex-4 LX-200 [8] (rys. 1). Na każdy układ FPGA przypada jeden moduł TIO (układ ASIC zapewniający komunikacje z resztą systemu poprzez magistralę NUMA-link) oraz pięć synchronicznych pamięci SRAM, zgrupowanych logicznie w trzy struktury.

3. Algorytm

Równie Schroedingera niezależne od czasu dla systemu składającego się z N cząstek będących we wzajemnej interakcji możemy wyrazić następującą zależnością [5]:

$$\hat{H}\psi = E\psi \tag{1}$$

gdzie:

ψ – funkcja falowa,
H – hamiltonianem,
E – energią układu.

Z równania (1) następnie uzyskuje się wybrane parametry badanego układu cząstek opisujące określone jego własności (np. całkowita energia). Błędy aproksymacji oraz obliczeń na etapie wyliczania funkcji falowej manifestują się następnie w dokładności uzyskanych wyników. Dlatego bardzo ważny jest dobór odpowiedniej precyzji obliczeń, która następnie pozwoli uzyskać wiarygodne wyniki. Z drugiej jednak strony w zbyt duża precyzja angażuje zwiększone zasoby logiczne układu FPGA.

W rzeczywistości liczba cząsteczek w badanym układzie jest na tyle duża, że nie istnieje możliwość znalezienia dokładnego rozwiązania analitycznie. W takim przypadku stosowane są przybliżenia, aby zredukować problem obliczeniowy. Omawiany w niniejszej pracy algorytm stanowi fragment procedury SCF (*Self-Consistent Field),* nazywanej czasem metodą Hartreego–Focka. Równanie Hartreego–Focka jest numerycznym przybli-

żaniem rozwiązania równania Schroedingera. Ogólna procedura rozwiązania równania Hartreego–Focka w zapisie macierzowym wyraża się następującym równaniem[5]:

$$FC - SCE = 0 \tag{2}$$

gdzie:

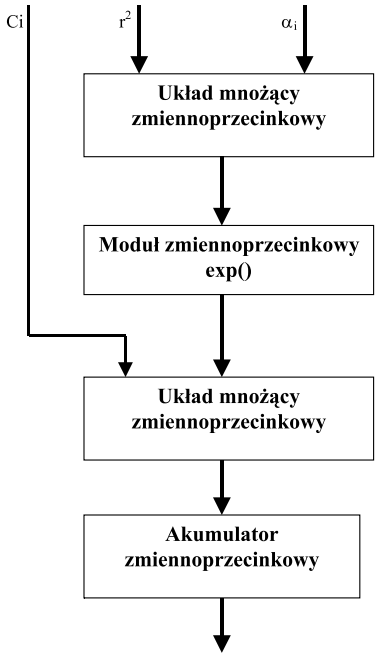
F – operator Focka,
C – macierz nieznanych współczynników,
S – macierz całek nakładania,
E – diagonalna macierz energii orbitalnych, wszystkie macierze są jednakowych rozmiarów.

Rozwiązanie równania Hartreego–Focka jest procesem iteracyjnym. W każdym kroku iteracji budowana jest macierz Focka, która zależy od orbitali atomowych (funkcji jednoelektoronowych). Dlatego orbitale muszą zostać wyliczone w każdej iteracji algorytmu. Eksponencjalna części funkcji orbitalnej w znaczący sposób wpływa na szybkość obliczeń i została zaimplementowana sprzętowo.

Cześć eksponencjalna funkcji orbitalnej jest wyrażona następującą zależnością:

$$\chi_r(r) = \sum_i c_i N_i e^{-\alpha_i r^2} = \sum_i C_i e^{-\alpha_i r^2} \tag{3}$$

Algorytm ten został podzielony na kilka sekcji, z których każda została zaimplementowana jako odrębny moduł (rys. 2).



Rys. 2. Schemat blokowy modułu EP, która realizuje obliczenia funkcji orbitalnej

Wszystkie bloki funkcjonalne, z których zbudowany jest moduł EP (moduł obliczający ekspotencjalną cześć orbitalu atomowego), pracują w pełni potokowo w celu zapewnienia jak najwyższej wydajności.

Układ mnożący zmiennoprzecinkowy

Moduł o modyfikowanej szerokości wejść, których wartość może się zmieniać w przedziale od pojedynczej do podwójnej precyzji. Architektura tej jednostki została omówiona dokładnie w [7].

Moduł zmiennoprzecinkowy exp()

W module zastosowano algorytm mieszany obliczania funkcji exp() (wielomianowo-tablicowy).

$$e^{x} = 2^{x\cdot\log_2 e} = 2^{x_i} \cdot e^{x - x_i / \log_2 e} \tag{4}$$

gdzie x_i jest częścią całkowitą $x\cdot\log_2 e$.

Architektura tej jednostki została omówiona dokładnie w osobnym dokumencie [6].

Akumulator zmiennoprzecinkowy

Jest to końcowy moduł logiki EP odpowiadający za sumowanie oraz przechowywanie wyników operacji wyliczania funkcji jednoelektronowej (orbitalu). Jest to jednostka w pełni potokowa i parametryzowana pracująca w standardzie zmiennoprzecinkowym w zakresie od pojedynczej do podwójnej precyzji.

4. Wyniki implementacji oraz uzyskane przyspieszenie obliczeń

Moduł EP został zaimplementowany na platformie RASC. Została zamieszczona ilość zajętych zasobów zajętych przez moduły sprzętowe dla dwóch skrajnych przypadków, pojedynczej oraz podwójnej precyzji.

Konstruktor platformy RASC firma SGI [1] dostarcza logikę *core services* stanowiącą interfejs pomiędzy algorytmem użytkownika oraz modułami zewnętrznymi podłączonymi do układu FPGA. Logika ta jest niezbędna do prawidłowej komunikacji oraz sterowania aplikacją umieszczoną w akceleratorze RASC, dlatego musi zostać uwzględniona w bilansie zajętości zasobów (tab. 1, 2).

Tabela 1
Wyniki implementacji modułu EP dla pojedynczej precyzji obliczeń

Wyniki implementacji	#4-input LUT	#FF	# 18-Kb BRAMs
Moduł EP	2229 (1%)	1975 (1%)	2 (0,006%)
Moduł EP + core services	11560 (7%)	15922 (9%)	25 (7%)

Tabela 2
Wyniki implementacji modułu EP dla podwójnej precyzji obliczeń

Wyniki implementacji	#4-input LUT	#FF	# 18-Kb BRAMs
Moduł EP	8684 (4,8%)	7891 (4%)	6 (0,02%)
Moduł EP + core services	18015 (10%)	21838 (12%)	29 (8%)

W ramach logiki EP moduł exp() jest jedynym, który wykorzystuje pamięci BRAM do przechowywania współczynników tablicowych (tab. 3).

Tabela 3
Wyniki implementacji modułów składowych logiki EP dla pojedynczej precyzji

Wyniki implementacji	#4-input LUT	#FF	# 18-Kb BRAMs
Exp()	820 (0,5%)	920 (0,5%)	2 (0,006%)
Ukł. mnożący	549 (0,3%)	444 (0,25%)	0
Akumulator	860 (0,5%)	605 (0,4%)	0

Tabela 4
Wyniki implementacji modułów składowych logiki EP dla podwójnej precyzji

Wyniki implementacji	#4-input LUT	#FF	# 18-Kb BRAMs
Exp()	5025 (3%)	5223 (3%)	6 (1,8%)
Ukł. mnożący	2316 (1%)	1850 (1%)	0
Akumulator	1343 (0,5%)	818 (0,4%)	0

Można zauważyć, że moduły Exp() oraz akumulator absorbują największą ilość zasobów logicznych (tab. 3, 4). Warto również zwrócić uwagę, że zaproponowane rozwiązanie odznacza się dość sporym opóźnieniem potokowym (tab. 5), które jest ceną za uzyskiwaną dużą szybkość przetwarzania danych.

Tabela 5
Opróżnienie potokowe poszczególnych modułów składowych

Moduł	Układ mnożący	Exp()	Akumulator	EP
Opóźnienie potokowe dla pojedynczej precyzji	4	21	8	33
Opóźnienie potokowe dla podwójnej precyzji	5	30	10	45

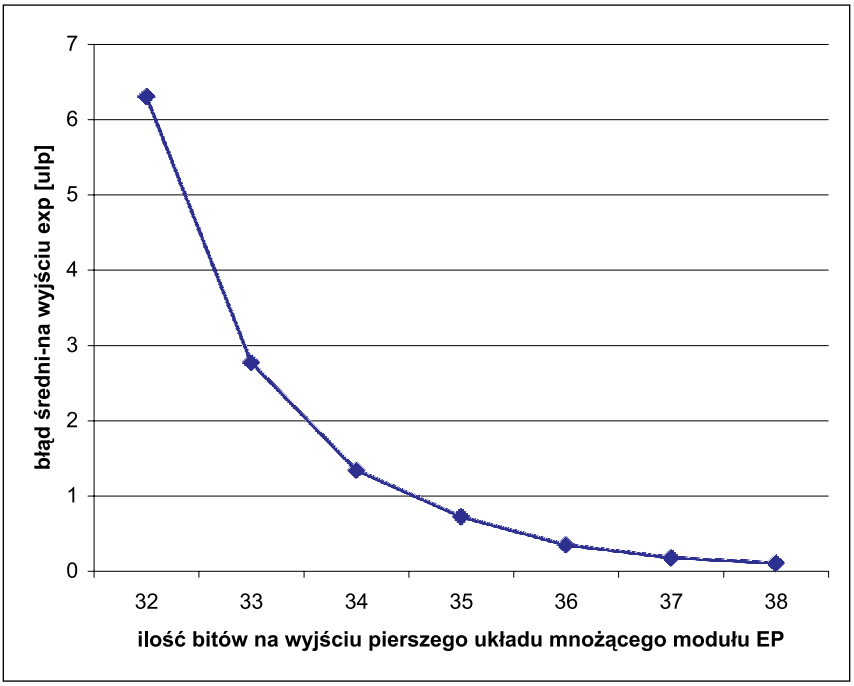
Ze względu na specyfikę algorytmu istnieje możliwość umieszczenia czterech jednostek EP pracujących standardzie pojedynczej precyzji obliczeń w jednym układzie FPGA na platformie RASC. Na podstawie przeprowadzonych pomiarów, obliczenia dla jednego 32-bitowego argumentu zajmują 4 ns łącznie z transferem danych w obu kierunkach dla częstotliwości pracy układu FPGA równej 200 MHz (tab. 6).

Tabela 6
Uzyskane przyspieszenie obliczeń

Czas wykonania pojedynczej operacji obliczania funkcji jednoelektrodowej przez procesor Itanium [ns]	Uzyskane przyspieszenie [RASC/Itanium]
20	5

W przedstawionej kalkulacji przyspieszania przyjęto założenie, że kod wykonywany przez procesor Itanium jest w pełni zoptymalizowany w sensie implementacji oraz opcji ustawień flag kompilatora.

Dla wszystkich możliwych połączeń w ramach projektowanego modułu można dokonywać odpowiedniego doboru ścieżki danych (rys. 3), preferowanym przez autora podejściem jest metoda góra – dół (*Top down*). Dysponując wiedzą, jaka jest wymagana wyjściowa precyzja nadrzędnego modułu, stopniowo dokonuje się analizy dokładności wyniku dla modułów znajdujących się stopień niżej w hierarchii projektu. Takie podejście umożliwia w efekcie określenie ilości bitów strażniczych (*Guard bits*) dla najbardziej elementarnych bloków składowych projektu. Oczywiście w tym miejscu pojawia się pytanie, w jaki sposób można się dowiedzieć, jaka jest graniczna precyzja obliczeń dla danego fragmentu przyspieszanej aplikacji, dla której wciąż otrzymywany jest poprawny wynik. Obliczanie naukowo-techniczne prowadzone są zazwyczaj w notacji zmiennoprzecinkowej podwójnej precyzji, gdyż jest to standardowy format zapisu danych stosowany w procesorach ogólnego przeznaczenia. Okazuje się jednak, że taka dokładność obliczeń jest często nadmiarowa i kosztowna w przypadku implementacji w układach FPGA, dlatego też stosowane są różne rozwiązania mające umożliwić określenie rzeczywistej wymaganej w danym algorytmie precyzji danych. Autor proponuje metodę zaburzania ścieżki danych aplikacji, której fragment ma być zaimplementowany w układach rekonfigurowalnych. Metoda ta polega na skalowaniu szerokości danych wejściowych aplikacji (fragmentu kodu wykonywany na procesorze) w celu określenia jej granicznej wartości precyzji obliczeń, przy której otrzymywany jest wciąż poprawny wynik. Należy nadmienić, że istnieje szereg bibliotek programistycznych umożliwiających definiowanie liczb o arbitralnie definiowalnej precyzji np. [9]. Wykorzystując tę metodę, udało się stwierdzić, że do zapewnienia poprawności obliczeń eksponencjalnej części orbitalu atomowego jak również samej funkcji orbitalu atomowego wystarczy pojedyncza precyzja obliczeń. Oryginalny algorytm wykonywany z zastosowaniem procesora ogólnego przeznaczenia pracuje na liczbach zmiennoprzecinkowych podwójnej precyzji.



Rys. 3. Zależność błędu średniego na wyjściu jednostki exp()
w zależności od szerokości bitowej pierwszego układu mnożącego (rys. 2)

5. Wnioski

Przedstawiona w niniejszej pracy implementacja stanowi kolejny krok do sprzętowej realizacji jednostki obliczającej potencjał korelacyjno-wymienny.

Wyniki implementacji obrazują niewielką ilość zajmowanych zasobów logicznych przez opisywany moduł, jest tak nawet dla podwójnej precyzji obliczeń. Uzyskano rzeczywiste przyspieszenie rzędu 5×, w porównaniu z realizacją tego samego algorytmu na procesorze Itanium 2 1,6 GHz. Należy zaznaczyć, że gdyby nie uwzględniać opóźnień wnoszonych przez transfer danych na magistrali komunikacyjnej pomiędzy procesorem i układem FPGA przyspieszenie byłoby 16×. Dlatego w systemie wieloprocesorowym, gdzie wymagany jest transfer danych pomiędzy procesorami w celu zrównoleglenia obliczeń, bardziej miarodajnym współczynnikiem przyśpieszenia byłby 16×.

Można oczekiwać, że, wyniki akceleracji będą znacznie lepsze w miarę zwiększania liczby stopni przetwarzania (stopnia skomplikowania algorytmu) zaimplementowanego w układzie FPGA, gdyż kwestie kosztów transferu danych będą odgrywały mniejszą rolę. Ważną kwestią jest również szerokość interfejsu komunikacyjnego układu FPGA, w przypadku platformy RASC wynosi ona 128 bitów. Autorzy dysponują obecnie również platfor-

mą firmy SRC Computers, której szerokość interfejsu komunikacyjnego wynosi 256, co automatycznie prowadzi do zwiększenia uzyskiwanego przyspieszenia do 10× bez żadnych dodatkowych modyfikacji struktury omawianego modułu. Należy oczekiwać, że producenci akceleratorów sprzętowych opartych o układy FPGA będą zwiększać szybkość interfejsów komunikacyjnych, gdyż takie postępowanie pozwala w pełni wykorzystać potencjał struktur logiki programowalnej.

Literatura

[1] Silicon Graphics, Inc., *Reconfigurable Application-Specific Computing User's Guide*. Ver. 005, January 2007, SGI.

[2] Giles M., *GPU's – the next big advance in HPC?* Reconfigurable Supercomputing Conference (MRSC), Belfast, Northern Ireland, April 1–3, 2008.

[3] Gothandaraman A., Peterson G., Warren G., Hinde R., Harrison R., *FPGA acceleration of a quantum Monte Carlo application*. Parallel Computing, 34(4–5), 2008, 278–291.

[4] Gothandaraman A., Warren G.L., Peterson G.D., Harrison R.J., *Reconfigurable accelerator for quantum Monte Carlo simulations in N-body systems*. Proceedings of the 2006 ACM/IEEE Conference on Supercomputing Tampa, Florida, November 11–17, 2006.

[5] Koch W., Holthausen M., *A Chemist's Guide to Density Functional Theory*. Wiley-VCH, 2 edition, Aug. 21, 2001.

[6] Wielgosz M., Jamro E., Wiatr K., *Highly Efficient Structure of 64-Bit Exponential Function Implemented in FPGAs*. ARC 2008, Lecture Notes in Springer-Verlag, London LNCS 4943, 274–279.

[7] Jamro E., Wielgosz M., Wiatr K., *Realizacja operacji mnożenia zmiennoprzecinkowego w układach FPGA*. Konferencja RUC'2009, w druku.

[8] *Xilinx Virtex-4 Family Overview* http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf.

[9] *C library for multiple-precision Floating point* http://www.mpfr.org/.