AUTOMATYKA • 2010 • Tom 14 • Zeszyt 3/2

Maciej Wielgosz*, Ernest Jamro*, Paweł Russek*, Kazimierz Wiatr*

Sprzętowa implementacja części wielomianowej funkcji orbitalnej na potrzeby obliczeń kwantowo-chemicznych

1. Wprowadzenie

Algorytmy stosowane w chemii kwantowej odznaczają się ogromnymi wymaganiami obliczeniowymi. W przypadku bardziej zaawansowanych układów cząsteczek, składających się z kilkunastu atomów, czas obliczeń realizowanych z wykorzystaniem komputerów dużej mocy jest rzędu kilku dni. Dlatego podejmowane były w przeszłości i są obecnie liczne inicjatywy mające na celu przyspieszenie tego rodzaju obliczeń [1, 2]. Zrodził się też pomysł wykorzystania układów FPGA do akceleracji obliczeń zmiennoprzecinkowych w chemii kwantowej. Matryce rekonfigurowalne wydają się atrakcyjne, ze względu na możliwości implementowania niskopoziomowych operacji, które stwarzają szerokie możliwości doboru oczekiwanej precyzji obliczeń. Ta własność ma szczególnie istotne znacznie w przypadku algorytmów iteracyjnych, dla których w kolejnych krokach oczekiwana jest, coraz większa dokładności obliczeń. Redukcja precyzji obliczeń prowadzi jednocześnie do zmniejszenia zajętości zasobów, a przede wszystkim do przyspieszenia wykonywania operacji. Układy FPGA dzięki silnie równoległej architekturze wewnętrznej pozwalają na dostosowanie struktury jednostki obliczeniowej do rozmiaru układu cząsteczek, dla jakiego prowadzone są obliczenia, uwzględniając typy powłok atomowych oraz ilości funkcji bazy.

Obecnie coraz powszechniej wykorzystywane są języki wysokiego poziomu takie jak Impuls C, Mitron C [8], które jednak pomimo skrócenia czasu projektowania, nie pozwalają na tak efektywne modelowanie bitowych operacji jak języki HDL. Dlatego postanowiliśmy w naszej pracy wykorzystać język VHDL, co zaowocowało również implementacją szeregu silnie zoptymalizowanych modułów zmiennoprzecinkowych, takich jak funkcja exp, akumulator, układ mnożący, które odznaczają się lepszymi parametrami niż odpowiadające im jednostki z biblioteki producentów [7].

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

Do realizacji prezentowanego w niniejszym artykule modułu została wykorzysta platforma SGI RASC, istnieje ona już kilka lat i zyskała sobie rzesze użytkowników. Pojedyncza karta zawiera dwa układy Xilinx Virtex-4 LX200 oraz 80 MB pamięci QDR SRAM [3]. Jest ona umieszczona w strukturze komputera SMP SGI Altix 4700. Autorzy przeprowadzili również badania wydajności wspomnianej platformy, opisane w osobnej pracy [4], co pozwoliło wyraźnie oddzielić wyniki prowadzonych badaniach od specyfiki i wydajności samej platformy.

2. Algorytm

Jedną z najbardziej popularnych i najprostszych technik aproksymacji równania Schroedingera jest metoda Hatree–Focka. Ogólną procedurą rozwiązania tego równania jest metoda samouzgodnienia. Wykonywana jest ona w sposób iteracyjny, aż do uzyskania oczekiwanego wyniku (oczekiwanej dokładności). Ogólna procedura rozwiązania równania Hartree–Focka w zapisie macierzowym wyraża się następującym równaniem [5]:

$$\boldsymbol{FC–SCE} = 0 \tag{1}$$

gdzie:

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

W niniejszym artykule prezentowany jest fragment algorytmu obliczania wartości orbitalu atomowego w punkcie bezpośrednio wykorzystywany w ramach metody DFT [5], który wyrażony jest następują formuła (2).

$$\chi_{klm}(r) = C_x \cdot C_y \cdot C_z \cdot r_x^k r_y^l r_z^m \sum_i c_i e^{-\alpha_i r^2} \tag{2}$$

gdzie wartości r_x, r_y, r_z są to współrzędne punktu gridu rzutowanego na atom, dla którego prowadzone są obliczenia, natomiast C_x, C_y, C_z są współczynnikami normalizacji. Wykładniki k, l, m zależą od typu powłoki atomowej (s, p, d lub f). Natomiast wartości C_i oraz α_i są współczynnikami bazy atomowej, w jakiej prowadzone są obliczenia. Zależność $r^2 = r_x^2 + r_y^2 + r_z^2$ określa położenie danego punktu rzutowego na każdy atom [5] w siatce przestrzennej.

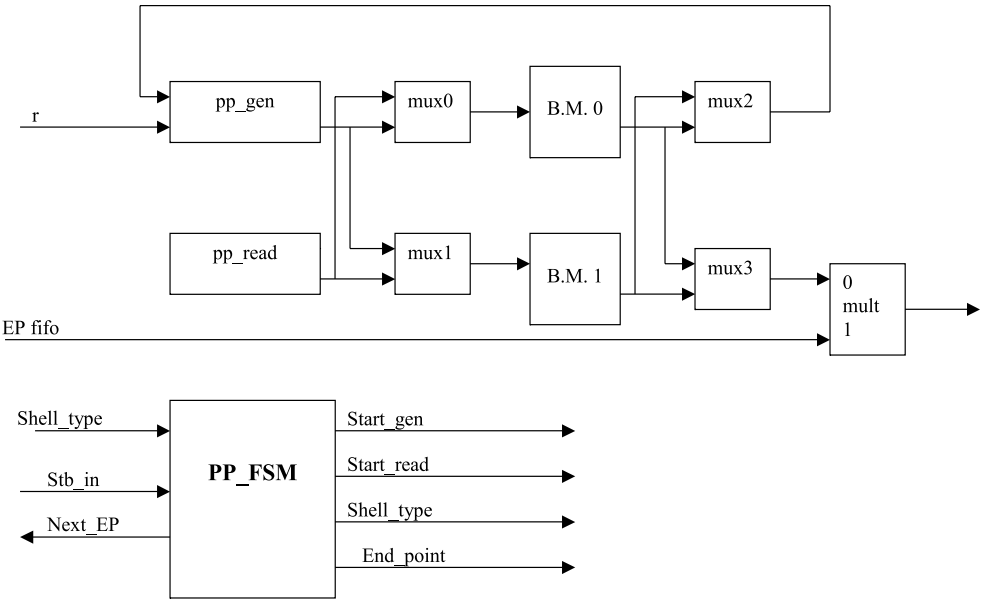
Obliczanie części wielomianowej funkcji orbitalnej wykonywane jest z wykorzystaniem współczynników eksponencjalnych otrzymanych jako wynik działania jednostki EP, opisanej w odrębnej publikacji [6]. Opisywane w tej pracy obliczenia prowadzone są

w oparciu o zależność wyrażoną równaniem (3), stanowi ona cześć składową formuły opisującej funkcję orbitalną (2).

$$\chi_w(r) = C_x \cdot C_y \cdot C_z \cdot r_x^k r_y^l r_z^m \tag{3}$$

3. Architektura modułu

Implementacja równania (3) z wykorzystaniem procesora ogólnego przeznaczenia sprowadza się do stworzenia struktury złożonej z kilku pętli obliczeniowych, w przeciwieństwie do implementacji w układach FPGA, która wymaga opracowania odrębnego sprzętowego modułu. Podstawowymi założeniami przyjętymi podczas implementacji algorytmu sprzętowej realizacji wielomianowej części orbitalu atomowego było wielokrotne wykorzystywanie raz obliczonych współczynników oraz generowanie ich z właściwym wyprzedzeniem, tak by możliwe było zapewnienie płynności pracy całego systemu obliczania funkcji orbitalnej. Z tego też względu moduł *PP* (obliczający część wielomianową orbitalu atomowego) podzielony został na dwa bloki składowe (rys. 1): *pp_gen*, *pp_read*, z których każdy spełnia odrębną funkcję. W zależności od typu powłoki równanie (3) przybiera różną postać (tab. 1), uwzględniającą również współczynniki normalizacji. Dla wyższych powłok rośnie ilość orbitali atomowych, co jak to zostanie w dalszej części artykułu pokazane wpływa na efektywność pracy modułu *PP*.

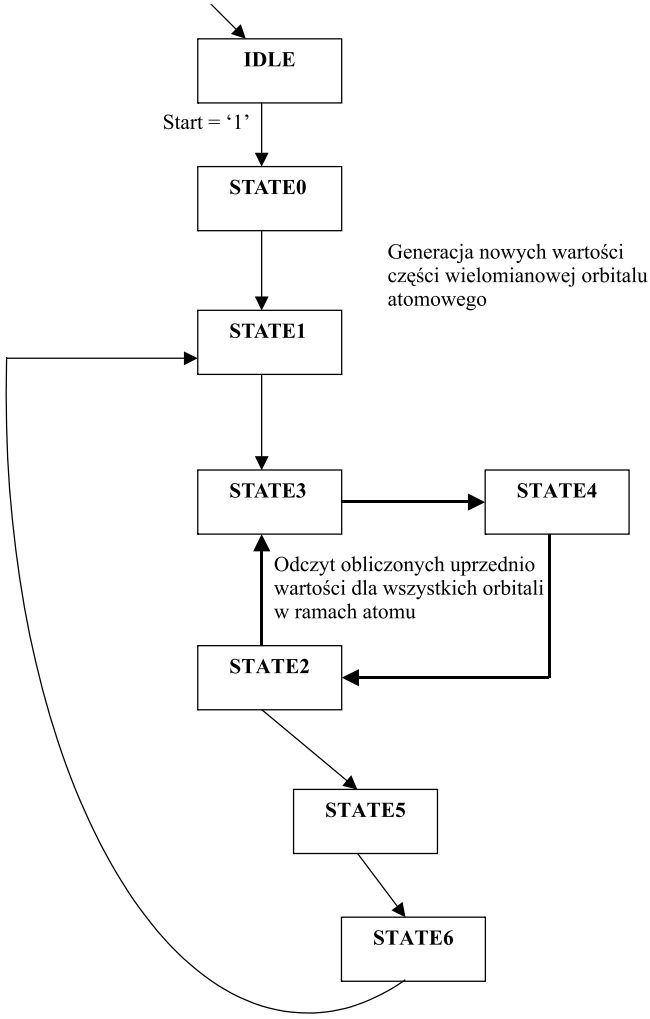


Rys. 1. Schemat blokowy logiki *PP* (obliczającej wielomianową część funkcji orbitalnej)

Tabela 1
Cześć wielomianowa orbitalu atomowego w zależności od typu powłoki

Typ powłoki	Typ orbitalu (wartości k,l,m)	Współczynnik normalizacji	Część wielomianowa
s	1	1	1
p	X	1	x
	Y	1	y
	Z	1	z
d	X^2	1/3	$1/3 \cdot x^2$
	xy	1	xy
	xz	1	xz
	Y^2	1/3	$1/3 \cdot y^2$
	yz	1	yz
	Z^2	1/3	$1/3 \cdot z^2$
f	X^3	2/30	$2/30 \cdot x^3$
	X^2y	1/3	$1/3 \cdot x^2y$
	X^2z	1/3	$1/3 \cdot x^2z$
	xy^2	1/3	$1/3 \cdot xy^2$
	xyz	1	xyz
	xz^2	1/3	$1/3 \cdot xz^2$
	Y^3	2/30	$2/30y^3$
	Y^2z	1/3	$1/3 \cdot y^2z$
	yz^2	1/3	$1/3 \cdot yz^2$
	Z^3	2/30	2/30

Logika *pp_gen* (rys. 1) generuje wartości współczynników wielomianowych funkcji orbitalnej, które będą używane w następnym kroku obliczeniowym. Natomiast jednostka *pp_read* odczytuje wartości obliczonych uprzednio (w poprzednim kroku) współczynników, na podstawie informacji o rodzaju powłoki, dla jakiej prowadzone są obliczenia. Układ mnożący wykorzystywany jest do otrzymania iloczynu części wielomianowej oraz eksponencjalnej [6]. Synchronizację pracy przedstawionych jednostek zapewnia automat stanu *pp_fsm* (rys. 2).



Rys. 2. Automat sterujący pracą logiki obliczającej cześć wielomianową orbitalu atomowego

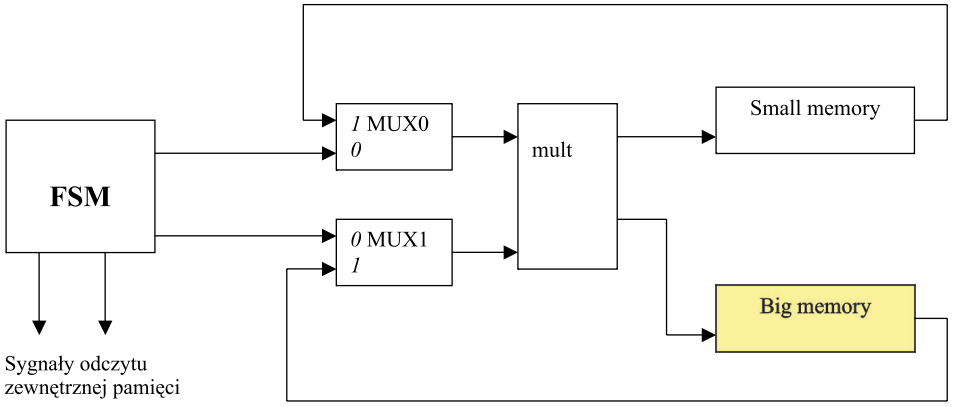
Moduł *pp_gen* odczytuje współrzędne punktów *rx*, *ry*, *rz* z pamięci FIFO i wykonuje obliczenia wszystkich części wielomianowych dla powłok s, p, d, f. Operacja ta wymaga 22 kroków obliczeniowych. Tak jak to zostało uwidocznione na schemacie blokowym modułu PP (rys. 1), pamięci wewnętrzne zostały zdublowane. Takie podejście umożliwia ich współdzielenie pomiędzy moduły *pp_gen* oraz *pp_read*, które naprzemiennie odczytują i zapisują ich zawartość.

Można zauważyć, że opisane rozwiązanie uwidacznia swoje zalety, gdy obliczenia prowadzone są dla powłok wyższych rzędów (p, d, f). W takiej sytuacji nakład obliczeniowy logiki *pp_gen* związany z wygenerowaniem współczynników wielomianowych

wykorzystywany jest w pełni podczas pracy modułu *pp_read*, gdyż raz wygenerowane wartości są następnie wielokrotnie wykorzystywane.

Natomiast gdy w obliczeniach pojawiają się dominująca ilość powłok typu s, nakład obliczeniowy związany z wygenerowaniem wszystkich orbitali dla danego zestawu *rx*, *ry*, *rz* nie jest wykorzystany. W takiej sytuacji moduł *pp_read* musi oczekiwać na logikę *pp_gen*. Proponowanym rozwiązaniem tej kwestii, a tym samym zapewnienie bardziej optymalnego wykorzystania zasobów obliczeniowych byłoby dokonywanie uprzedniego rozpoznania, jaka jest najwyższa powłoka dla danego punktu i atomu (punktu rzutowanego na atomie) i następnie podanie tej informacji do automatu sterującego logiki *pp_gen*, tak by generować współczynniki tylko dla rzeczywiście istniejących powłok. Z wprowadzeniem wspomnianej modyfikacji wiązałoby się jednak opóźnienie w uruchomieniu jednostki *pp_gen*, związane z uprzednim określeniem najwyższej powłoki atomowej w danym cyklu obliczeniowym, co w znaczącym stopniu zwiększyłoby ilość operacji sekwencyjnych w wykonywanych w ramach modułu. W konsekwencji rozwiązanie to nie zostało wprowadzone.

Schemat blokowy modułu *pp_gen* został zamieszczony na rysunku 3, można zauważyć, że moduł ten zwiera w swojej strukturze dwa rodzaje pamięci, z których tylko jedna *big memory* jest dostępna zewnętrznie (współdzielona) oraz zawiera dane wykorzystywane w dalszych obliczeniach. Pamięć *small memory* pełni natomiast rolę pomocniczą przechowując chwilowe wartości, które są wykorzystywane w procesie obliczania współczynników wielomianowych umieszczanych następnie w pamięci *big memory*. Skąd następnie współczynniki te czytane są i wymnażanie przez uprzednio obliczone wartości eksponencjalne otrzymane jako rezultat działania modułu EP [6].



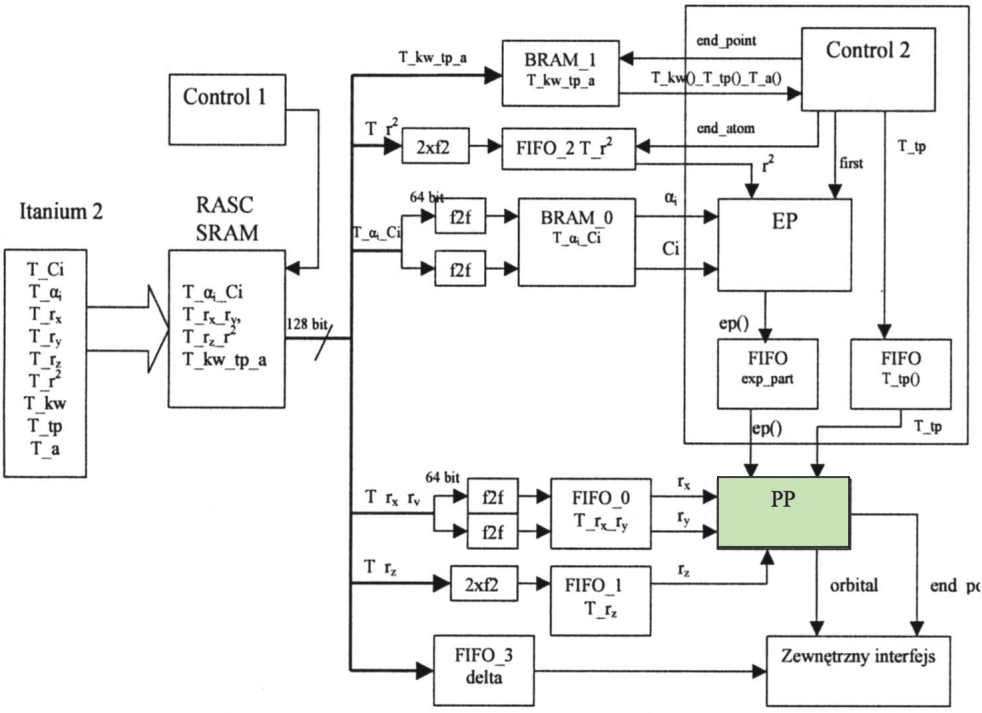
Rys. 3. Automat sterujący pracą logiki obliczającej część wielomianową orbitalu atomowego

Należy zaznaczyć, że pamięć *small memory* zaimplementowana jest w strukturze FPGA jako distributed memory ze względu na niedużą ilość przechowywanych współczynników pomocniczych (tylko sześć). Pełen cykl obliczania wszystkich współczynników wie-

lomianowych orbitali od s do f wymaga przejścia dwudziestu p stanów automatu stanu logiki *pp_gen*. Automat synchronizuje pracę wewnętrznych pamięci BRAM, multiplekserów oraz pobierania danych wejściowych. Dzięki wielokrotnemu wykorzystaniu obliczonych wartości pośrednich możliwa była znaczna redukcja ilości stanów wewnętrznych automatu oraz czasu generowania współczynników wielomianowych. Ze względu na ilość danych przechowywanych i konieczność korzystania z dwóch interfejsów w przypadku pamięci *big memory* zastosowano BRAM. Zapewnia ona również komunikację pomiędzy modułami *pp_gen* oraz *pp_read*.

4. Osadzenie jednostki *PP* w strukturze modułu *Orbital*

Po zaimplementowaniu jednostka PP została umieszczona w systemie obliczającym wartość orbitalu atomowego w punkcie (rys. 4). Moduł ten złożony jest z szeregu bloków logicznych realizujących obliczenia oraz zapewniających komunikację z procesorem Itanium 2 1.6 GHz. Procesor ten spełnia rolę gospodarza w systemie obliczeniowym i steruje pracą akceleratora, realizując jednocześnie obliczenia części sekwencyjnej algorytmu generowania macierzy potencjału korelacyjno-wymiennego [5].



Rys. 4. Osadzenie jednostki *PP* w strukturze modułu *Orbital*

5. Wyniki implementacji

Poniżej została przedstawiona informacja dotycząca zajętości zasobów logicznych układu FPGA dla dwóch skrajnych przypadków szerokości magistrali danych modułu obliczającego cześć wielomianową orbitalu atomowego. Dla modułu pracującego w standardzie zmiennoprzecinkowym podwójnej oraz pojedynczej precyzji, w którym zajętość zasobów jest wielokrotnie mniejsza (tab. 2 i 3). Należy nadmienić, że precyzja pojedyncza została wybrana, jako docelowa, gdyż jest wystarczająca w przypadku metody DFT [5].

Tabela 2
Wyniki implementacji modułu *PP* dla podwójnej precyzji obliczeń

Wyniki implementacji	#4-input LUT	#FF	# 18-Kb BRAMs
Moduł PP	4164 (2%)	6257 (3%)	6 (0.06%)
Moduł PP+core services	13495 (11%)	20204 (11%)	29 (8%)

Tabela 3
Wyniki implementacji modułu *PP* dla pojedynczej precyzji obliczeń

Wyniki implementacji	#4-input LUT	#FF	# 18-Kb BRAMs
Moduł PP	1499 (0.6%)	1180 (0.5%)	2 (0.02%)
Moduł PP + core services	10830 (6%)	15127 (8%)	25 (7%)

Czas wykonania operacji *PP* wynosi średnio 10 taktów zegara, od momentu podania sygnału odczytu do chwili pojawiania się pierwszego współczynnika orbitalu na wyjściu. Należy zaznaczyć, że generowanie pierwszego zestawu współczynników wielomianowych zachodzi w czasie, gdy nie pojawiły się jeszcze pierwsze współczynniki eksponencjalne. Jest to wynikiem opóźnienia potokowego modułu *EP* [6].

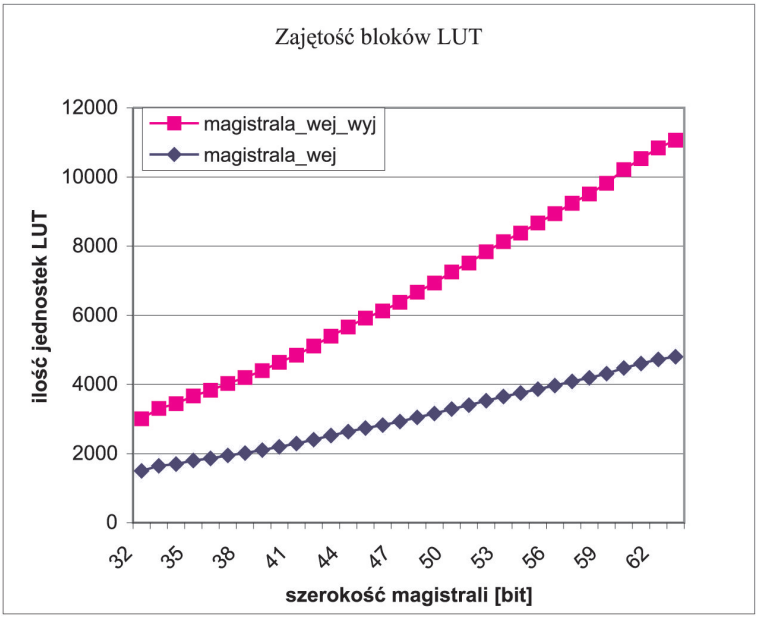
Prezentowany wykres (rys. 5) obrazują przyrost zasobów logicznych LUT modułu *PP* dla dwóch przepadków:

- Gdy zmianie ulega tylko szerokość magistrali wejściowej danych (szerokość magistrali wyjściowej pozostaje stała).
- Gdy modyfikowana jest liczba bitów zarówno magistrali wejściowej jak i wyjściowego interfejsu danych (w zakresie od 32 do 64 bitów).

Na podstawie powyższych zależności można zauważyć, że zmiana szerokości magistrali wejściowej w znacznie mniejszym stopniu wpływa na zajętość zasobów układu programowalnego niż modyfikacja zarówno wejściowej i wyjściowej szyny danych. Dlatego też w sytuacji, gdy pobierane dane są podwójnej, a dalsze obliczenia (w kolejnych krokach) prowadzone będą w pojedynczej precyzji, zalecane jest na etapie obliczeń części wielo-

mianowej zredukować precyzję wyniku, gdyż przyczyni się to znacząco do zmniejszenia wykorzystania zasobów przez logikę PP.

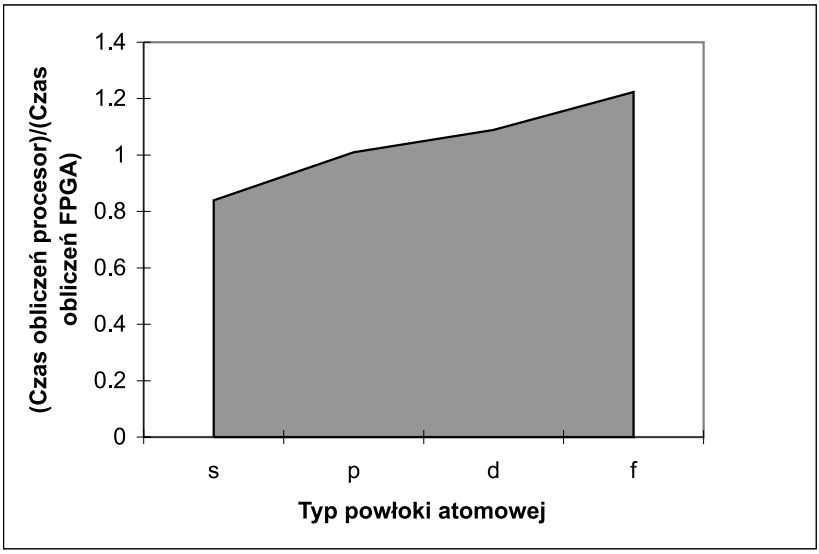
Przeprowadzone zostały porównawcze testy obliczeniowe dla cząsteczki wody. W tym celu dokonano pomiaru czasu wykonania algorytmu obliczania orbitalu atomowego na procesorze Itanium 2 1,6 GHz oraz na platformie RASC. Dla bloku 512 punktów czas ten wynosi 2885 us, natomiast czas wykonania tych samych obliczeń na platformie RASC to około 3174 us.



Rys. 5. Zajętość bloków *LUT* modułu *PP* zaimplementowanego w układzie *Xilinx Virtex-4 LX200* w zależności od szerokości magistrali danych

Należy zauważyć, że w przypadku cząsteczki wody dominująca jest powłoka typu s. Fakt ten znacząco wpływa na uzyskiwaną efektywność sprzętowego wykonania algorytmu z wykorzystaniem akceleratora sprzętowego, gdyż niewykorzystywane są uprzednio obliczone wartości części wielomianowych dla wyższych powłok. Natomiast wykonanie tej operacji na procesorze wymaga tylko jednej iteracji pętli i jest mało absorbujące obliczeniowo.

Wraz ze wzrostem rozmiaru powłok atomowych, dla których prowadzone są obliczenia, uwidaczniają się korzyści płynące z zastosowania akceleratora sprzętowego, gdyż ilość iteracji wykonania operacji na procesorze wzrasta przy jednoczesnym zwiększeniu wykorzystania mechanizmów tablicowania oraz potokowości w układzie FPGA, które można uznać za znaczący przyczynek do wzrostu efektywności obliczeniowej. Na wykresie (rys. 6) zobrazowana została opisywana zależność.



Rys. 6. Uzyskana akceleracja obliczania funkcji orbitalnej w zależności od typu powłoki, dla jakiej prowadzone są obliczenia

6. Wnioski

Zaprezentowana w niniejszej pracy implementacja stanowi etap do pełnej sprzętowej implementacji jednostki obliczającej potencjał korelacyjno-wymienny. W ramach zrealizowanego projektu powstał szereg jednostek zmiennoprzecinkowych, które odznaczają się niewielką zajętością zasobów oraz parametryzowaną szerokością magistral komunikacyjnych. Zaprojektowany moduł pracuje w sposób potokowy, wykorzystując pojedynczy układ mnożący. Uzyskiwane przyspieszenie zależy od charakteru cząsteczki, dla jakiej prowadzone są obliczenia, a w szczególności od typu powłok atomowych. W kolejnym kroku wykonana zostanie w pełni sprzętowa realizacja funkcji obliczającej wartość orbitalu atomowego w punkcie.

Literatura

[1] 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.

[2] Ramdas T., Egan G., Abramson D., Baldridge K., *Towards a special-purpose computer for Hartree-Fock computations What's on the table*, *and how do we take it?* Theoretica Chimica Acta, vol. 120, No. 1–3/May, 2008, 138.

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

[4] Wielgosz M., Jamro E., Wiatr K., *Accelerating calculations on the RASC platform. A case study of the exponential function*. Applied Reconfigurable Computing, ARC'2009, Springer-Verlag, LNCS 5453, 2009, 306–311.
[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., *Implementacja w układach FPGA modułu obliczającego funkcję jednoelektronową*. Automatyka (półrocznik AGH), t. 13, z. 3, 2009, 1043–1050.
[7] Strona firmy Xilinx zawierająca informację odnośnie rozwiązań typu IP-core http://www.xilinx.com/ipcenter/index.htm.
[8] Środowisko projektowania dla systemów HPRC: Mitrion-c, http://www.mitrionics.com/.