

Od algorytmu dynamicznej cieczy sieciowej do dedykowanego komputera równoległego II – maszyna mDLL

Streszczenie. W artykule opisano złożenia projektowe, budowę i realizację maszyny przeznaczonej do symulacji zjawisk zachodzących w wieloskładnikowych układach molekularnych. Przedstawiony system elektroniczny zbudowano przy użyciu programowalnych układów scalonych FPGA (w j. ang. Field Programmable Gate Array). W maszynie zaimplementowano model dynamicznej cieczy sieciowej (ang. dynamic lattice liquid - DLL) i wykonano testy jej działania.

Abstract. The article describes the assumption of design, construction and implementation of a machine intended to simulate the phenomena occurring in complex molecular systems. Presented electronic system was built with the Field Programmable Gate Array (FPGA). In the machine was implemented a model of Dynamic Lattice Liquid (DLL) and the tests of the device operation was performed. (From dynamic lattice liquid algorithm to dedicated parallel computer II – mDLL machine).

Słowa kluczowe: programowalne układy FPGA, topologia sieci połączeń, równoległe przetwarzanie danych, symulacje molekularne.

Keywords: Field Programmable Gate Array, topology of the network connections, parallel data processing, molecular simulations.

Wstęp

Rozbudowane systemy obliczeniowe takie jak klastry komputerowe lub superkomputery bardzo często używane są do symulacji zjawisk zachodzących w złożonych układach molekularnych [1,2,3]. Jednak możliwości jakie dają symulacje komputerowe podlegają bardzo istotnym ograniczeniom. Jednym z nich jest niewielki, w stosunku do potrzeb, rozmiar przestrzeni symulacyjnej. Dla mniejszych systemów ograniczenie to wynika ze skończonych zasobów pamięci oraz ilości jednocześnie pracujących jednostek obliczeniowych, a dla superkomputerów umożliwiających badanie obiektów zawierających dużą liczbę elementów ograniczenie jest związane z możliwością przeprowadzenia symulacji w „rozsądnym czasie” - to znaczy nie przekraczającym czasu trwania od kilku do kilkunastu miesięcy ciągłej pracy systemu.

W pracy przedstawiono budowę skalowalnej maszyny mDLL zawierającej komórki operacyjne (KDLL) [4] umieszczone w węzłach powierzchniowo centrowanej sieci przestrzennej (w j. ang. face-centered cubic – FCC) tworzącej torus trójwymiarowy [5]. Urządzenie przeznaczone jest do wykonywania symulacji molekularnych z wykorzystaniem algorytmu dynamiki cieczy sieciowej (w j. ang. dynamic lattice liquid - DLL) [6,7]. mDLL stanowi rozwinięcie koncepcji maszyny DLL zbudowanej wcześniej w Politechnice Łódzkiej.

W maszynie DLL opisanej w pracach [8,9,10], na 6 płytach PCB (w j. ang. Printed Circuit Board) umieszczono po 3 układy FPGA (w j. ang. field-programmable gate array), w których znajdowało się po 12 komórek operacyjnych KDLL (Rys. 1B). Komórki KDLL zaimplementowane w układach FPGA znajdowały się na wirtualnych płaszczyznach w węzłach sieci trójkątnej o liczbie koordynacyjnej 6 (rys. 1A). Dla każdej zaimplementowanej komórki KDLL przyporządkowano po 6 wyprowadzeń ułożonych na płytach PCB, a pozostałe połączenia (2 razy po 3 połączenia do komórek KDLL umieszczonych na sąsiednich płaszczyznach) zrealizowano za pomocą kabli (rys. 1A i rys. 1C). Wraz z układami sterującymi pracą maszyna DLL stanowiła zamkniętą strukturę przestrzenną.

Zastosowane w maszynie DLL rozwiązanie topologiczne posiadało jednak istotne wady:

- 1) maszyna była nieskalowana – nie można było jej rozbudowywać o większą liczbę komórek KDLL,
- 2) przyjęty system transmisji danych pomiędzy komórkami

KDLL, ze względu na ograniczoną liczbę końcówek lutowniczych w układach FPGA uniemożliwiał umieszczenie więcej niż kilkunastu komórek KDLL w jednym układzie FPGA,

- 3) konieczna była konfiguracja układów FPGA na trzy sposoby. Wynikało to z odmiennej topologii linii wejściowo-wyjściowych dla 3 sąsiadujących ze sobą płaszczyzn. Każda z płaszczyzn przesunięta była względem sąsiedniej o 1/2 węzła sieci trójkątnej (przedstawienie płaszczyzn P1, P2 i P3 na rysunkach 1A lub 1C).

Na podstawie wykonanych badań nad maszyną DLL i analizy jej przyszłych zastosowań wysnuto trzy kluczowe wnioski stanowiące podstawę do podjęcia działań mających na celu zbudowanie nowego urządzenia (o roboczej nazwie mDLL). Wnioski były następujące:

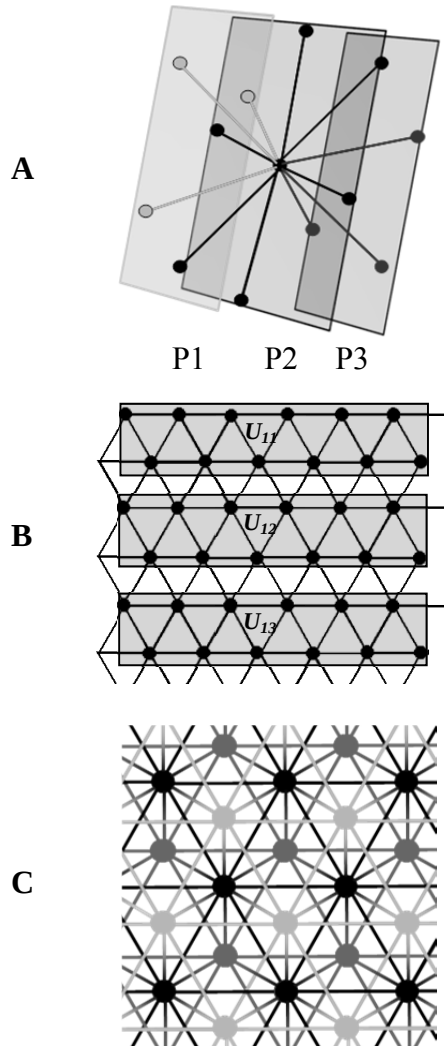
- 1) należy zaprojektować skalowalną maszynę mDLL zawierającą takie same jak w maszynie DLL komórki operacyjne KDLL umieszczone w węzłach sieci FCC w układzie torusa trójwymiarowego.
- 2) należy opracować oraz testować nowy system wymiany danych pomiędzy sąsiadującymi płytami PCB zawierającymi układy FPGA, ponieważ w najnowszych układach FPGA może być implementowanych co najmniej kilkadziesiąt komórek operacyjnych KDLL.
- 3) W celu sprawdzenia nowego systemu wymiany danych maszyna mDLL powinna zawierać co najmniej jeden skalowalny moduł otoczony 26 podobnymi modułami, płyty PCB muszą być tak mocowane przestrzennie, aby można było w przyszłości wytworzyć urządzenie, które będzie zawierało co najmniej 1 milion komórek KDLL niezbędnych do wykonania symulacji dynamiki cząsteczek w złożonych cieczach molekularnych.

Struktura przestrzenna maszyny mDLL

Prace nad budową maszyny mDLL rozpoczęto od analizy topologii sieci FCC oraz systemu wymiany danych pomiędzy komórkami KDLL. Przyjęto, że płaszczyzny będą przecinać węzły w inny sposób niż to przedstawiono na rysunku 1. Tym razem węzły sieci FCC stanowiły co drugi węzeł zredukowanej sieci kubicznej ZSK, dla której odległość pomiędzy węzłami wynosiła $ax/2$, a sieć FCC była jakby „zanurzona” w sieci ZSK (rys. 2A).

Z analizy fragmentu sieci FCC przedstawionej na rysunku 2A wynika, że można wyróżnić dwa rodzaje równoległych płaszczyzn, usytuowanych względem siebie naprzemiennie, na których znajdują się węzły sieci FCC. Pierwsza płaszczyzna (nazywana dalej *płaszczyzną prostą*)

rozpięta jest na 5 węzłach o współrzędnych $(i-1, j+1, k)$, $(i+1, j+1, k)$, $(i+1, j-1, k)$, $(i-1, j-1, k)$ oraz (i, j, k) . Drugi rodzaj płaszczyzny (nazywanej dalej *płaszczyzną obróconą*) rozpięty jest na 4 węzłach o współrzędnych $(i, j+1, k-1)$, $(i+1, j, k-1)$, $(i, j-1, k-1)$ i $(i-1, j, k-1)$ (taka sama płaszczyzna zawiera punkty $(i, j+1, k+1)$, $(i+1, j, k+1)$, $(i, j-1, k+1)$ i $(i-1, j, k+1)$). Na rysunkach 2B i 2C przedstawiona jest różnica pomiędzy *płaszczyznami prostą* i *obróconą* .

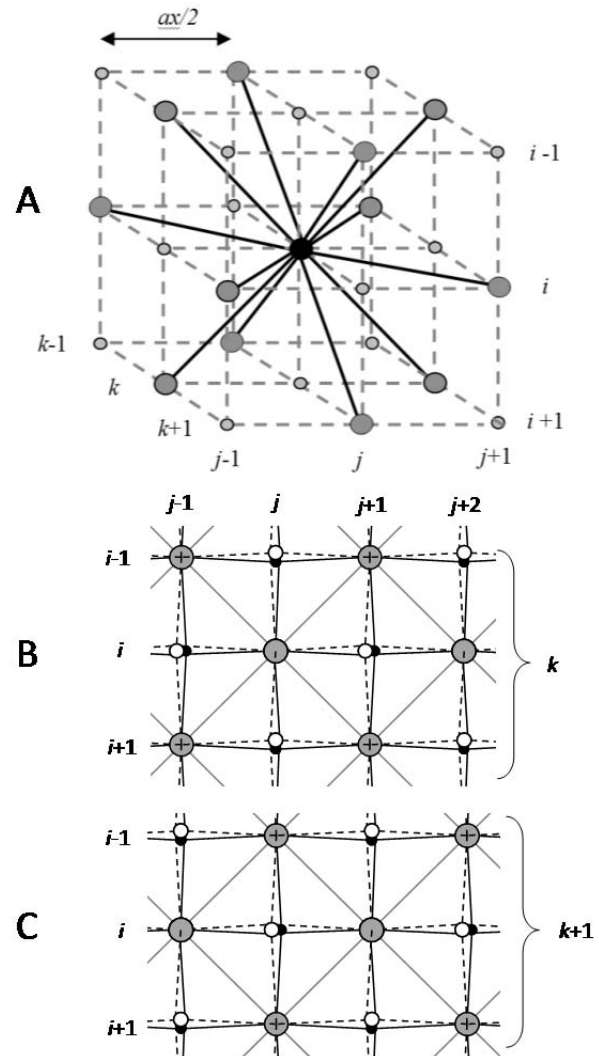


Rys. 1. Płaszczyzny dla sieci FCC, w których umieszczono komórki operacyjne KDLL w maszynie DLL wraz z siecią połączeń. Płaszczyzny, w których znajdowali się najbliżsi sąsiedzi komórki KDLL (A); węzły sieci leżące w jednej płaszczyźnie, zawarte w 3 układach skalonych U_{11} , U_{12} i U_{13} (B); połączenia pomiędzy komórkami KDLL (C). Kolorami oznaczono płaszczyzny: P1 - ciemnoszara „za kartką”, P2 - czarna „w płaszczyźnie kartki” i P3 - jasnoszara „przed kartką” (C)

Z rysunków 2B i 2C wynika, że *płaszczyzna obrócona* jest to *płaszczyzna prosta* przesunięta o jeden węzeł sieci zredukowanej ZSK. W dalszej części pracy *płaszczyzną prostą* będziemy nazywali płaszczyznę, w której pierwszy punkt w lewym, górnym rogu ($i=1$ i $j=1$) zawierać będzie węzeł sieci FCC. W przeciwnym przypadku płaszczyzna nazywana będzie *płaszczyzną obróconą* .

Komórkom KDLL w każdym układzie FPGA, przypisano położenia w co drugim spośród $\alpha \cdot \beta \cdot \eta$ węzłów wirtualnej (zaimplementowanej w układzie FPGA) sieci ZSK (rys. 3). Dla parzystej liczby węzłów sieci liczba komórek zawartych w układzie logicznym wynosiła $\alpha\beta\eta/2$, a gdy liczba ta była

nieparzysta, to liczba tych komórek wynosiła $(\alpha\beta\eta/2)-1$ lub $(\alpha\beta\eta/2)+1$.



Rys. 2. Zredukowana sieć kubiczna ZSK (A); *płaszczyzny prosta* (B); i *obrócona* (C). Indeksami i, j, k oznaczono położenie węzłów w sieci ZSK.

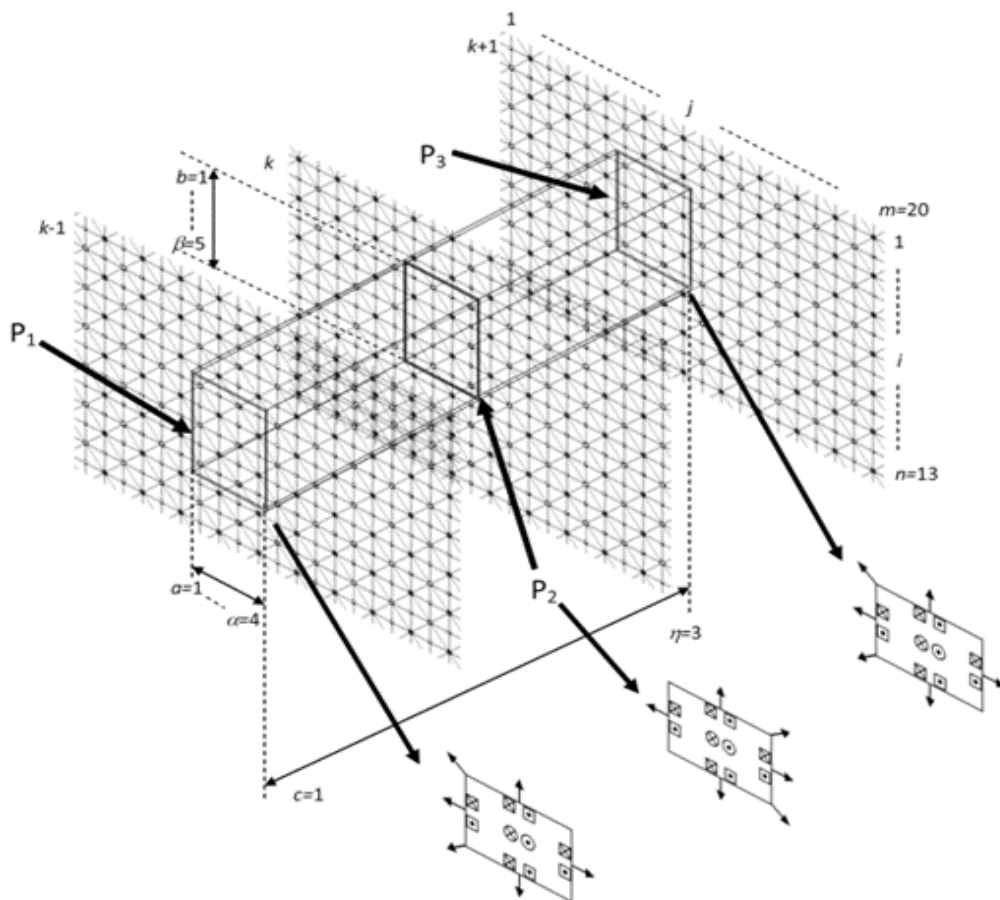
W celu implementacji komórek operacyjnych KDLL w układach FPGA należało zdefiniować topologię sieci ZSK wpisanej do tych układów. Trzeba było też wyznaczyć liczbę linii potrzebnych do wymiany danych pomiędzy sąsiadującymi ze sobą układami FPGA w określonych kierunkach przestrzeni. W zależności od tego czy liczby wierszy β i kolumn α były parzyste czy nieparzyste, rozróżniono 4 pary układów *komplementarnych* różniących się pomiędzy sobą kierunkami komunikacji [9,11]. Układy te ułożone były naprzemiennie na płaszczyźnie i stanowiły pary nazwane dalej *komplementarnymi*, w których układy oznaczone literą P nazywane zostały *podstawowymi*, a układy U nazywane zostały *uzupełniającymi*.

Na rysunku 3 przedstawiono przykładowe rozmieszczenie 3 fragmentów płaszczyzn: dwóch *prostych* i jednej *obróconej* objętych przez jeden układ FPGA zawierający 30 komórek operacyjnych znajdujących się w węzłach zredukowanej sieci ZSK. Układ zawiera $\alpha=4$ kolumn, $\beta=5$ wierszy i $\eta=3$ płaszczyzn. Na płaszczyznach oznaczonych indeksami $k-1$ oraz $k+1$ znajdują się warstwy P_1 i P_3 układu FPGA, w których kierunki komunikacji z sąsiednimi komórkami w tych płaszczyznach odpowiadają układowi *podstawowemu* dla α parzystego i β nieparzystego.

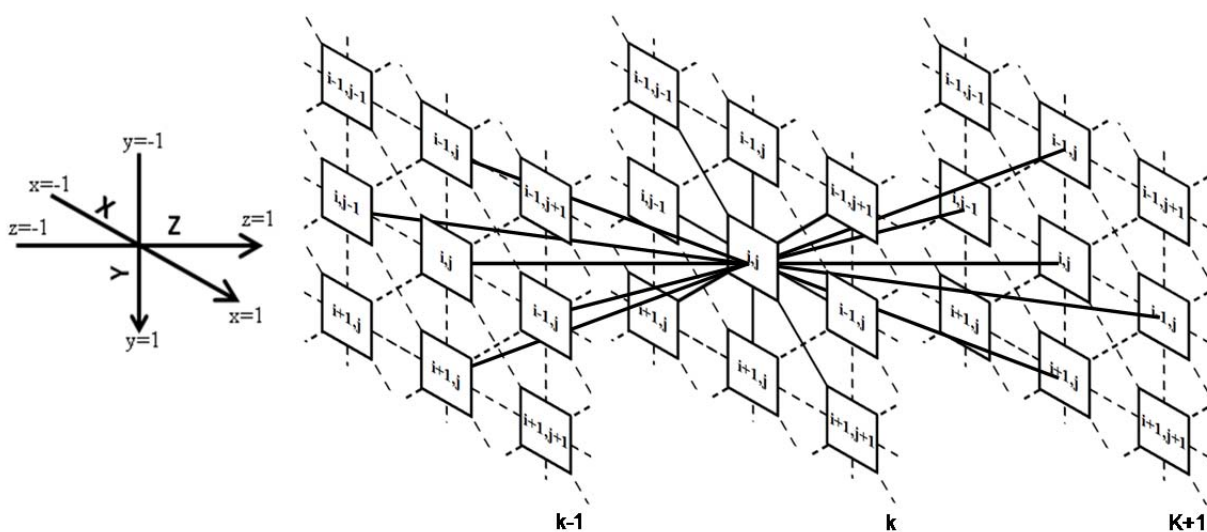
W trzecim wymiarze (wzdłuż osi z), podobnie jak dla dwóch wymiarów (w płaszczyźnie x,y), występują 4 możliwości wzajemnego ułożenia płaszczyzn (zależne od parzystości α i β), w których występują pary płaszczyzn *prostych* i *obróconych*. Dla każdej kombinacji par płaszczyzn, naprzeciw siebie zawsze znajdują się

komplementarne układy podstawowy i uzupełniający (Tabela 1).

W maszynie mDLL komórki operacyjne KDLL znajdujące się na ścianach sieci zaimplementowanej w układzie FPGA komunikują się z sąsiadami w 18 kierunkach za pomocą zewnętrznych linii sygnałowych (Rys. 4).



Rys. 3. Rozmieszczenie 3 fragmentów płaszczyzn: dwóch *prostych* i jednej *obróconej* o wymiarach $\alpha=4$, $\beta=5$ i $\eta=3$, objętych przez układ FPGA (linie ciągłe) zawierający 30 komórek operacyjnych znajdujących się w węzłach zredukowanej sieci ZSK



Rys. 4. Kierunki wymiany danych dla jednego układu FPGA. Wstawka wyznacza orientację osi współrzędnych w przestrzeni

Tabela 1

α	β	Płaszczyzna prosta	Płaszczyzna obrócona
nieparzyste	nieparzyste		
parzyste	nieparzyste		
nieparzyste	parzyste		
parzyste	parzyste		

W wybranym kierunku liczba tych linii, oznaczona symbolem $\Omega_{\alpha,\beta,\eta}^{X,Y,Z}_{x,y,z}$, wynosi:

$$(1) \quad \Omega_{\alpha,\beta,\eta}^{X,Y,Z}_{x,y,z}(i, j, k) = \Pi \cdot \Theta$$

gdzie: X, Y, Z wyznaczają kierunki przestrzenne i przyjmują wartości binarne (np. $X=1$ i $\bar{X}=0$ – dane wysyłane, albo $X=0$ i $\bar{X}=1$ – dane nie wysyłane w kierunku osi X),
– x, y i z (zmiennie dwuwartościowe -1 albo 1) informują w którą stronę wzdłuż kierunków X, Y i Z wysyłane/odbierane są dane (patrz wstawka widoczna na rysunku 4),
– indeksy i, j to numery wiersza i kolumny, a indeks k to numer płaszczyzny, w której znajduje się układ FPGA (rys. 3),
– Ω jest równe 1 gdy urządzenie jest maszyną prostą (układ FPGA znajduje się w lewym górnym rogu pierwszej płaszczyzny dla $i=1, j=1$ oraz $k=1$) lub Ω jest równe -1 dla maszyny obróconej (gdy lewy górny róg pierwszej płaszczyzny nie zawiera układu logicznego).
– α, β, η oznaczają liczbę węzłów sieci ZSK zawartych w układzie FPGA liczonych w kierunkach osi x, y i z (rys. 3).

Sześciowymiarowy, wierszowy wektor Π definiuje kierunki w przestrzeni i dany jest wzorem:

$$\Pi = [\bar{X}\bar{Y}\bar{Z} \quad \bar{X}Y\bar{Z} \quad \bar{X}\bar{Y}Z \quad XY\bar{Z} \quad \bar{X}YZ \quad X\bar{Y}Z]$$

a sześciowymiarowy, kolumnowy wektor Θ , którego wyrazy zależą od liczby węzłów (α, β, η) sieci ZSK zawartych w

układzie FPGA i od parzystości parametrów α, β i η opisany jest równaniem:

$$\Theta = \begin{bmatrix} 2\eta\beta - \eta - \beta \\ 2\alpha\eta - \alpha - \eta \\ 2\alpha\beta - \alpha - \beta \\ \frac{1}{2} \left[\eta - \Omega_{\alpha,\beta,\eta}^{X,Y,Z}_{x,y,z} \left(\underset{\leftarrow}{\beta} \underset{\leftarrow}{i} + \underset{\leftarrow}{y} \underset{\rightarrow}{\beta} \right) \left(\underset{\leftarrow}{\alpha} \underset{\leftarrow}{j} + \underset{\leftarrow}{x} \underset{\rightarrow}{\alpha} \right) \right] \\ \frac{1}{2} \left[\alpha - \Omega_{\alpha,\beta,\eta}^{X,Y,Z}_{x,y,z} \left(\underset{\leftarrow}{\eta} \underset{\leftarrow}{k} + \underset{\leftarrow}{z} \underset{\rightarrow}{\eta} \right) \left(\underset{\leftarrow}{\beta} \underset{\leftarrow}{i} + \underset{\leftarrow}{y} \underset{\rightarrow}{\beta} \right) \right] \\ \frac{1}{2} \left[\beta - \Omega_{\alpha,\beta,\eta}^{X,Y,Z}_{x,y,z} \left(\underset{\leftarrow}{\alpha} \underset{\leftarrow}{j} + \underset{\leftarrow}{x} \underset{\rightarrow}{\alpha} \right) \left(\underset{\leftarrow}{\eta} \underset{\leftarrow}{k} + \underset{\leftarrow}{z} \underset{\rightarrow}{\eta} \right) \right] \end{bmatrix}$$

przy czym zmienne podkreślone strzałkami: skierowanymi: w lewo, w prawo lub w dwie strony odpowiadające liczbom naturalnym s , przyjmują wartości -1, 0 i 1 zgodnie z poniższą definicją:

$$(2) \quad \underset{\leftarrow}{s} = \frac{1 - (-1)^s}{2}; \quad \underset{\rightarrow}{s} = \frac{1 + (-1)^s}{2}; \quad \leftrightarrow s = (-1)^s$$

Maszyny DLL i mDLL zaprojektowano tak, że układy FPGA jednocześnie odbierają i wysyłają dane do sąsiadów i dlatego każdemu kierunkowi przyporządkowane są dwie linie sygnałowe. Zatem liczba wyprowadzeń z każdego układu scalonego FPGA (liczba końcówek lutowicznych) potrzebnych do połączenia komórek KDŁA z sąsiednimi komórkami w innych układach FPGA na sąsiednich płytach PCBM wynosi $2 \cdot \Omega_{\alpha,\beta,\eta}^{X,Y,Z}_{x,y,z}$. Równanie (1) pozwala wyznaczyć liczbę tych wyprowadzeń. W tabeli 2 przedstawiono wyniki obliczeń dla 12 komórek zaimplementowanych w jednym układzie FPGA ($\alpha=12, \beta=2, \eta=1$) (tak jak to zrealizowano w maszynie DLL opisaną w pracach [8,9]) oraz dla przypadku 100 zaimplementowanych komórek w jednym układzie.

Z danych zamieszczonych w tabeli 2 wynika, że liczba wyprowadzeń oraz liczba przewodów sygnałowych przypadających na jeden układ FPGA zawierający 12 komórek umożliwia wykonanie maszyny o topologii połączeń podobnej do maszyny DLL [8,9]. Jeśli jednak liczba zaimplementowanych komórek osiągnie wartość 100 lub więcej, to wtedy nie będzie możliwa budowa maszyny, w której komórki będą ze sobą połączone bezpośrednio za pomocą linii sygnałowych. Jest to spowodowane dwoma czynnikami:

- 1) liczba potrzebnych wyprowadzeń byłaby większa niż liczba dostępnych końcówek lutowicznych w jednym układzie scalonym (np. dostępnych w układach FPGA bloków OSERDES i ISERDES, które pozwalają na transmisję w standardzie LVDS (w j. ang. Low-Voltage Differential Signaling)),
- 2) liczba przewodów rozprowadzonych pomiędzy płytami byłaby zbyt wielka, aby maszyna mogła działać na zadowalającym poziomie niezawodności.

Ze wzoru (1) wynika, że liczba linii komunikacyjnych zależy od liczby węzłów ZRK implementowanych w układach FPGA. Jednak, ze względu na skalowalność maszyny mDLL, liczba przewodów potrzebnych do przesyłania sygnałów powinna być zawsze taka sama, niezależnie od liczby komunikujących się komórek KDŁA zaimplementowanych w układach FPGA. Uzyskano to poprzez kompensację danych i redukcję 18 kierunków

wyznaczonych przez linie komunikacyjne (rys. 4) do 6 kierunków wyznaczonych liniami sieci kubicznej. Dane gromadzono w pakiety, a następnie przesyłano je do sąsiednich układów FPGA szybkimi szeregowymi łączami za pośrednictwem 6 dwukierunkowych kanałów

komunikacyjnych (rys. 5). Szczegółowy opis sposobu gromadzenia danych w pakiety oraz ich wysyłania i odbierania czytelnik może znaleźć w dokumencie [11].

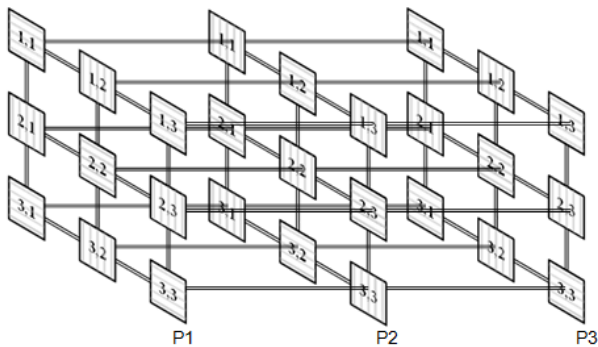
Tabela 2

Kierunek wymiany danych	Zwrot kanałów komunikacyjnych	Liczba linii kierunkowych dla układu FPGA	
		$\alpha \cdot \beta \cdot \eta / 2 = 12$	$\alpha \cdot \beta \cdot \eta / 2 = 100$
$X\bar{Y}\bar{Z}$	$x = 1$	${}_{12,2,1}^1 L_{1,y,z}^{1,0,0}(3,3,3) = 1$	${}_{5,5,8}^1 L_{1,y,z}^{1,0,0}(3,3,3) = 67$
	$x = -1$	${}_{12,2,1}^1 L_{-1,y,z}^{1,0,0}(3,3,3) = 1$	${}_{5,5,8}^1 L_{-1,y,z}^{1,0,0}(3,3,3) = 67$
$\bar{X}Y\bar{Z}$	$y = 1$	${}_{12,2,1}^1 L_{x,1,z}^{1,0,0}(3,3,3) = 11$	${}_{5,5,8}^1 L_{x,1,z}^{1,0,0}(3,3,3) = 67$
	$y = -1$	${}_{12,2,1}^1 L_{x,-1,z}^{1,0,0}(3,3,3) = 11$	${}_{5,5,8}^1 L_{x,-1,z}^{1,0,0}(3,3,3) = 67$
$\bar{X}\bar{Y}Z$	$z = 1$	${}_{12,2,1}^1 L_{x,y,1}^{0,0,1}(3,3,3) = 34$	${}_{5,5,8}^1 L_{x,y,1}^{0,0,1}(3,3,3) = 40$
	$z = -1$	${}_{12,2,1}^1 L_{x,y,-1}^{0,0,1}(3,3,3) = 34$	${}_{5,5,8}^1 L_{x,y,-1}^{0,0,1}(3,3,3) = 40$
XYZ	$x = 1, y = -1$	${}_{12,2,1}^1 L_{1,-1,z}^{1,1,0}(3,3,3) = 0$	${}_{5,5,8}^1 L_{1,-1,z}^{1,1,0}(3,3,3) = 4$
	$x = -1, y = 1$	${}_{12,2,1}^1 L_{-1,1,z}^{1,1,0}(3,3,3) = 0$	${}_{5,5,8}^1 L_{-1,1,z}^{1,1,0}(3,3,3) = 4$
	$x = 1, y = 1$	${}_{12,2,1}^1 L_{1,1,z}^{1,1,0}(3,3,3) = 1$	${}_{5,5,8}^1 L_{1,1,z}^{1,1,0}(3,3,3) = 4$
	$x = -1, y = -1$	${}_{12,2,1}^1 L_{-1,-1,z}^{1,1,0}(3,3,3) = 1$	${}_{5,5,8}^1 L_{-1,-1,z}^{1,1,0}(3,3,3) = 4$
$\bar{X}YZ$	$y = 1, z = -1$	${}_{12,2,1}^1 L_{x,1,-1}^{0,1,1}(3,3,3) = 6$	${}_{5,5,8}^1 L_{x,1,-1}^{0,1,1}(3,3,3) = 3$
	$y = -1, z = 1$	${}_{12,2,1}^1 L_{x,-1,1}^{0,1,1}(3,3,3) = 6$	${}_{5,5,8}^1 L_{x,-1,1}^{0,1,1}(3,3,3) = 2$
	$y = 1, z = 1$	${}_{12,2,1}^1 L_{x,1,1}^{0,1,1}(3,3,3) = 6$	${}_{5,5,8}^1 L_{x,1,1}^{0,1,1}(3,3,3) = 2$
	$y = -1, z = -1$	${}_{12,2,1}^1 L_{x,-1,-1}^{0,1,1}(3,3,3) = 6$	${}_{5,5,8}^1 L_{x,-1,-1}^{0,1,1}(3,3,3) = 3$
$X\bar{Y}Z$	$x = 1, z = -1$	${}_{12,2,1}^1 L_{1,y,-1}^{1,0,1}(3,3,3) = 1$	${}_{5,5,8}^1 L_{1,y,-1}^{1,0,1}(3,3,3) = 3$
	$x = -1, z = 1$	${}_{12,2,1}^1 L_{-1,y,1}^{1,0,1}(3,3,3) = 1$	${}_{5,5,8}^1 L_{-1,y,1}^{1,0,1}(3,3,3) = 2$
	$x = 1, z = 1$	${}_{12,2,1}^1 L_{1,y,1}^{1,0,1}(3,3,3) = 1$	${}_{5,5,8}^1 L_{1,y,1}^{1,0,1}(3,3,3) = 2$
	$x = -1, z = -1$	${}_{12,2,1}^1 L_{-1,y,-1}^{1,0,1}(3,3,3) = 1$	${}_{5,5,8}^1 L_{-1,y,-1}^{1,0,1}(3,3,3) = 3$
Razem	Końcówek lutowicznych (np. standard LVDS)	$2 \cdot {}_{12,2,1}^1 L_{x,y,z}^{X,Y,Z}(3,3,3) = 228$	$2 \cdot {}_{5,5,8}^1 L_{x,y,z}^{X,Y,Z}(3,3,3) = 774$
	Przewodów pomiędzy panelami	$2 \cdot {}_{12,2,1}^1 L_{x,y,z}^{X,Y,1}(3,3,3) = 96$	$2 \cdot {}_{5,5,8}^1 L_{x,y,z}^{X,Y,1}(3,3,3) = 248$

Kolejnym zagadnieniem było opracowanie odpowiedniej topologii ułożenia układów FPGA. W symulacjach molekularnych obszar przestrzenny dostępny dla cząsteczek w maszynie jest ograniczony, a w większości przypadków analizowane obiekty zawierają wielką liczbę cząsteczek i chcemy, aby granice układu znajdowały się w nieskończoności. W rzeczywistych układach zbudowanie sieci o nieskończenie wielkich rozmiarach nie jest możliwe i dlatego w maszynach równoległych, w których wielka liczba komórek operacyjnych systemu (np. mikroprocesorów) działa współbieżnie w czasie, stosowane są n-wymiarowe

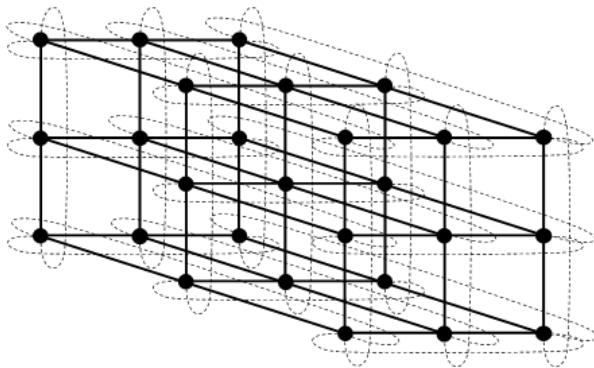
pseudonieskończone sieci [1213,14]. W maszynie DLL [8,9,10] komórki KDLL umieszczono w węzłach trójwymiarowego torusa, który powstaje po połączeniu ze sobą skrajnych elementów sieci (rys. 6). Jednak takie rozwiązanie uniemożliwia budowanie maszyny skalowalnej, ponieważ dokładanie kolejnych układów FPGA spowoduje, że długość połączeń skrajnych elementów sieci rośnie i dla dużych systemów może ona przekraczać kilka metrów. Połączenia te będą wtedy stanowiły najwolniejszy element całej maszyny czyniąc ją zbyt wolną do wykonywania

symulacji w obiektach zawierających wielką liczbę elementów (np. 1 mln lub więcej).



Rys. 5. Zredukowane kanały komunikacyjne pomiędzy układami FPGA

W opisywanej maszynie mDLL skalowalność uzyskano dzięki zastosowaniu techniki nazywanej w literaturze „leap frog” [5], która polega na naprzemiennym łączeniu układów scalonych znajdujących się w węzłach sieci torusa trójwymiarowego. Na rynku istnieją obudowy w postaci metalowych szaf przystosowane do takiego łączenia ze sobą płyt PCB zawierających układy scalone lub mikroprocesory. Jednak wykorzystanie takich szaf do budowy maszyny mDLL naruszałoby jej konstrukcję przestrzenną, w której płyty PCB z układami FPGA z układami FPGA umocowane są na płaskich, pionowo usytuowanych panelach. Opracowano więc inny sposób rozmieszczenia układów FPGA, który nie zmienia struktury przestrzennej maszyny i jednocześnie wykorzystuje system naprzemiennego łączenia układów scalonych (szczegóły tego rozwiązania znajdują się w opisach patentowych [15, 16]).



Rys. 6. Sieć kubiczna w kształcie trójwymiarowego torusa

Założenia konstrukcyjne maszyny mDLL

Maszynę mDLL zbudowano według następujących założeń:

- w układach FPGA zaimplementowano wirtualną zredukowaną sieć kubiczną ZSK o liczbie węzłów $\alpha \cdot \beta \cdot \eta$ (przy czym $\alpha = \beta$), a w co drugim węźle tej sieci umieszczono komórki operacyjne KDLL,
- położenia komórek KDLL w sieci ZSK skonfigurowanej w układach FPGA wyznaczono za pomocą indeksów a, b, c (indeks $a=1 \dots \alpha$ odpowiada współrzędnej x , indeks $b=1 \dots \alpha$ odpowiada współrzędnej y , a indeks $c=1 \dots \eta$ odpowiada współrzędnej z),
- urządzenie składa się z $\delta=3$ paneli zawierających $n=3$ wierszy (liczonych w kierunku osi y) i $m=3$ kolumn (liczonych w kierunku osi x),
- numery paneli oznaczono indeksem k , przy czym panel o numerze $k=1$ stanowi płaszczyznę prostą,

- na panelach, na przecięciu każdej kolumny (numerowanej indeksem i) i każdego wiersza (numerowanego indeksem j) umieszczono płyty PCB zawierające po 4 układy FPGA „przeniesione” z węzłów: i, j ; $i, 2m-j+1$; $2n-i+1, j$ i $2n-i+1, j$ wirtualnej sieci kubicznej. Węzły tej sieci obejmują obszary zajmowane przez $\alpha \cdot \beta$ węzłów ZSK zaimplementowanych w poszczególnych układach FPGA. Obszary te stanowią podsieć pełnej sieci ZSK całego panelu o $2m \cdot \alpha$ wierszach i $2n \cdot \beta$ kolumnach [17]. Cztery układy FPGA należące do jednego panelu oznaczono, odpowiednio, indeksami $s=1, s=2, s=3, s=4$.

Dla tak zdefiniowanej maszyny sformułowano szereg równań pozwalających opisać jej funkcjonalność i konstrukcję przestrzenną. Najbardziej użyteczny w praktyce okazał się układ równań służący do przeliczania wirtualnych współrzędnych przestrzennych x_s, y_s, z_s komórek operacyjnych zaimplementowanych w układach FPGA na ich współrzędne x_s, y_s, z_s w rzeczywistej sieci ZSK.

Równanie służące do wyliczania współrzędnych x_s, y_s, z_s komórki operacyjnej KDLL, której położenie w układzie FPGA wyznaczały indeksy a, b, c , umieszczonej w pozycji s na płycie PCB oznaczonej indeksami i, j znajdującej się na k -tym panelu ma następującą postać:

$$(3) \quad (x, y, z) = \omega \cdot \Gamma \cdot (x_s, y_s, z_s)$$

gdzie wartości współrzędnych x_s, y_s, z_s wynoszą:

$$\begin{aligned} x_s &= \lambda_{11}^s [(j-1) \cdot \alpha + a] + \lambda_{12}^s (2m+1) \\ y_s &= \lambda_{21}^s [(i-1) \cdot \alpha + b] + \lambda_{22}^s (2n+1), (k-1) \\ z_s &= (k-1)\beta + \lambda_{31}^s c + \lambda_{32}^s (\eta+1) \end{aligned}$$

Elementy macierzy λ_{pq}^s mogą przyjmować wartości 0, 1 lub -1 i wyrażone są za pomocą wzoru:

$$\begin{bmatrix} \lambda_{11}^s & \lambda_{12}^s \\ \lambda_{21}^s & \lambda_{22}^s \\ \lambda_{31}^s & \lambda_{32}^s \end{bmatrix} = \begin{bmatrix} (s-2) \cdot \left(\overleftarrow{s-s} \right) - \overleftarrow{s} & \frac{1}{2} \left(\overrightarrow{4s+2s} \overleftarrow{s-s-s} \right) \\ \overleftarrow{2s-s+3s} & \frac{1}{2} \left(\overleftarrow{s-s-2s} \right) \\ \overleftarrow{s-s} & \overrightarrow{s} \end{bmatrix}$$

gdzie s podkreślone strzałkami definiuje wzór (2).

Wielkość Γ jest równa 0 gdy w węźle sieci ZSK nie ma komórki operacyjnej, a równa 1, gdy komórka zajmuje miejsce w węźle. Γ wyraża się wzorem:

$$\begin{aligned} \Gamma &= \eta \cdot \left[\overrightarrow{\alpha \cdot (A+B+C)} + \overleftarrow{\alpha \cdot (A+B+C+I+J)} \right] \\ &+ \eta \cdot \left[\overleftarrow{\alpha \cdot (A+B+C+k)} + \overrightarrow{\alpha \cdot (A+B+C+I+J+k)} \right] \end{aligned}$$

Zmienne A, B, C, I, J odpowiadają indeksom a, b, c, i, j przeliczonym z układu z cyklicznymi warunkami brzegowymi do układu otwartego (sześciannu zawierającego sieć punktów ZSK) i wynoszą odpowiednio:

$$\begin{aligned} A &= \lambda_{11}^s a + \lambda_{12}^s (\alpha + 1) & J &= \lambda_{11}^s j + \lambda_{12}^s (2m + 1) \\ B &= \lambda_{21}^s b + \lambda_{22}^s (\alpha + 1) & I &= \lambda_{21}^s i + \lambda_{22}^s (2n + 1) \\ C &= \lambda_{31}^s c + \lambda_{32}^s (\eta + 1) \end{aligned}$$

Współczynnik ω jest równy:

$$\omega = \frac{\eta + \delta}{1 + (\delta + \eta)}$$

i wynosi 0 gdy η i δ są nieparzyste (niemożliwe jest wtedy spełnienie cyklicznych warunków brzegowych). W pozostałych przypadkach $\omega \neq 1$.

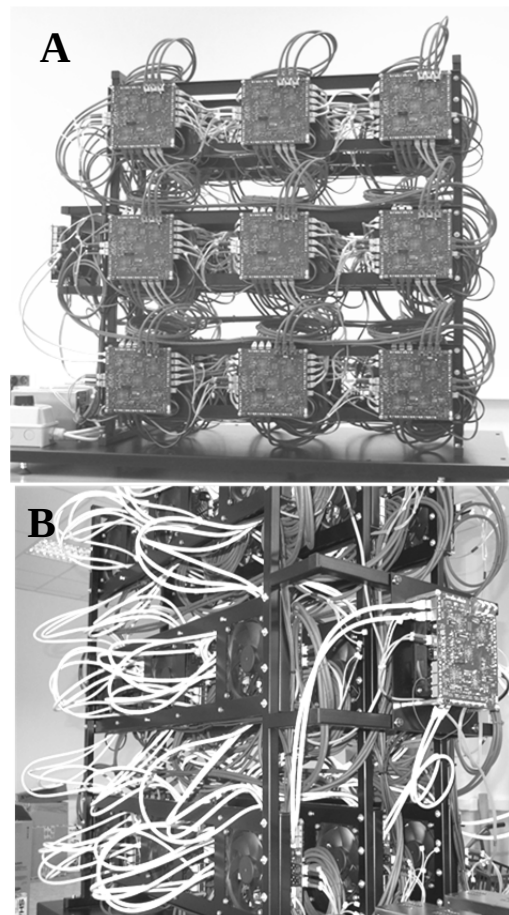
Wzór (3) umożliwia określenie początkowego położenia wszystkich elementów symulowanego układu oraz śledzenie trajektorii ich ruchu.

Realizacja elektroniczna

Urządzenie mDLL wykonano w taki sposób, aby można było sprawdzić opracowane koncepcje skalowalności maszyny i sposobu przesyłania informacji pomiędzy sąsiadującymi komórkami operacyjnymi. Maszyna mDLL z umocowanymi płytami PCB na stalowej kratownicy i przewodami do przesyłania danych widoczna jest na rysunku 7. Składa się ona z 3 pionowych, równoległych do siebie ułożonych paneli, na których równomiernie rozmieszczono po 9 płyt PCB. Płyta PCB umiejscowiona w centralnej części środkowego panelu otoczona była podobnymi 26 płytami. Przy takiej konstrukcji przestrzennej wymiana danych pomiędzy układami FPGA należącymi do centralnej płyty i układami znajdującymi się na sąsiednich płytach musiała się odbywać tylko za pośrednictwem fizycznie zamontowanych łącz sygnałowych. Pozwoliło to testować opracowaną koncepcję szeregowego przesyłania danych. W 108 układach FPGA zaimplementowano sieci ZSK zawierające po 32 komórki KDLL co dało 3240 komórek KDLL w całej maszynie. Wszystkie układy zasilane były za pośrednictwem przetwornic napięciowych, które chłodzone były powietrzem za pomocą wentylatorów. Płyta z dodatkowym układem FPGA służącym do koordynacji jednoczesnej pracy wszystkich komórek (układ sterujący maszyną USDLL [4,8,9]) znajdowała się na bocznej ścianie urządzenia (rys. 7B).

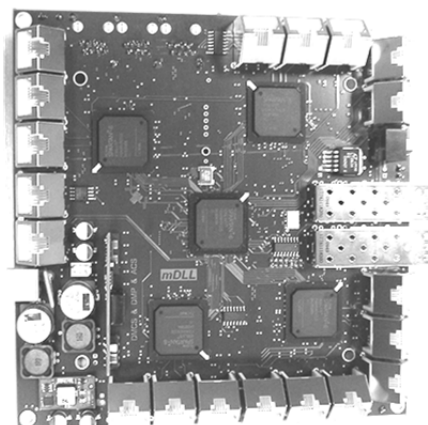
Wymiana danych pomiędzy poszczególnymi modułami maszyny odbywała się na 3 sposoby:

- 1) w celu minimalizacji czasu trwania transmisji, wynikającej z jej szeregowego charakteru, do wysyłania i odbierania danych pomiędzy komórkami operacyjnymi zastosowano transmisję w standardzie LVDS z szybkością do 800 Mb/s. Do połączenia układów należących do różnych płyt PCB zastosowano kable Ethernet. W każdym z 6 kanałów komunikacyjnych znajdowało się 6 „skrętek”, co pozwoliło wymieniać dane pomiędzy komórkami z prędkością 3,2 Gb/s.
- 2) dla potrzeb synchronizacji pracy maszyny, wymagającej komunikacji pomiędzy komórkami operacyjnymi i układem sterującym CKDLL, zastosowano mechanizm przesyłania sygnałów za pośrednictwem połączonych szeregowo płyt PCB. Do tego celu użyto dwóch kabli Ethernet, czyli 8 skrętek, z czego 5 skrętek zostało poświęconych na komunikację w kierunku „do CKDLL”, a 3 skrętki zostały przeznaczone na komunikację w kierunku „od CKDLL”.
- 3) do celów konfiguracji układów FPGA, inicjalizacji pracy maszyny i akwizycji wyników symulacji zastosowano linie światłowodowe oraz dostępne w układach Spartan 6 bloki MGT (w *j. ang.* Multi-Gigabit Transceiver) wspierające transmisję po tych liniach z prędkością do 3,125 Gb/s w każdym kierunku.



Rys. 7. Maszyna mDLL zawierająca 27 płyt PCB umieszczonych na 3 panelach. Na zdjęciu oznaczonym literą B widoczny jest układ sterujący maszyną USDLL

Na każdej z płyt PCB umieszczono po 5 układów FPGA (o numerze katalogowym XC6SLX75-FGG484-2), z których jeden znajdował się w ich centralnym miejscu i służył jako element zarządzający pracą pozostałych czterech (zdjęcie na rys. 8).



Rys. 8. Płyta PCB z układami FPGA. Centralny układ służył do synchronizacji pracy pozostałych 4, w których zaimplementowano komórki KDLL.

Układ ten został wyposażony w pamięć konfiguracji EEPROM (w *j. ang.* Electrically Erasable Programmable Read-Only Memory). Jej zawartość w momencie załączenia zasilania służyła do konfigurowania układu centralnego w taki sposób aby mógł on komunikować się z innymi

podobnymi znajdującymi się na sąsiednich płytach. Utworzono łańcuch szeregowo połączonych układów centralnych, z których pierwszy był z układem CKDLL. Dzięki takiemu rozwiązaniu z poziomu komputera PC połączonego z CKDLL, z wykorzystaniem protokołu JTAG, skonfigurowano wszystkie układy FPGA w maszynie.

Testy działania maszyny mDLL

Ważnym elementem weryfikującym poprawność działania maszyny mDLL było sprawdzenie mechanizmów komunikacji pomiędzy układami FPGA. Dokonano tego za pomocą dwóch testów.

Pierwszy test polegał na autoweryfikacji sąsiadów, z którymi każdy z układów FPGA się komunikuje. Każdy transponder dowolnego układu FPGA wysyłał unikalny identyfikator, jednoznacznie określający położenie i kierunek nadawania tego transpondera. Po stronie odbiorczej otrzymane identyfikatory były porównywane z wartościami oczekiwanymi a wszelkie odstępstwa były raportowane jako błędy połączeń.

Drugi test przeprowadzono analizując ruch wirtualnej „kulki” poruszającej się przypadkowo pomiędzy komórkami KDLL znajdującymi się w węzłach sieci FCC wewnątrz trójwymiarowego torusa. W czasie tego testu zapamiętywana była liczba odwiedzin „kulki” w każdym węzle, a oczekiwanym wynikiem był równomierny rozkład tychże odwiedzin we wszystkich węzłach sieci. Jakość tego rozkładu była oceniana poprzez ocenę rozrzutu liczby odwiedzin φ wyliczanej według wzoru:

$$(4) \quad \varphi = \frac{n_{\max} - n_{\min}}{n_{\max}}$$

gdzie $n_{\max}$ i $n_{\min}$ to największa i najmniejsza zaobserwowana liczba odwiedzin w dowolnym z węzłów.

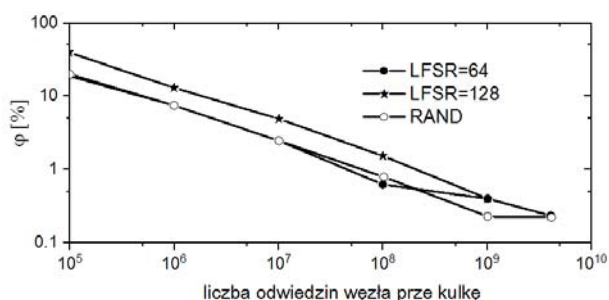
Wyniki przeprowadzonych testów przedstawiono na rysunku 9. Wynika z nich, że φ maleje wraz ze wzrostem liczby kroków algorytmu, co jednoznacznie dowodzi, że wszystkie połączenia pomiędzy węzłami zaimplementowanej sieci FCC były wykorzystywane w takim samym stopniu.

Test „kulki” był też znakomitym sposobem weryfikacji zastosowanego generatora liczb pseudolosowych. Został on przeprowadzony dla różnych szerokości rejestru LFSR (w j. ang Local Feedback Shift Register), a otrzymane wyniki zostały dodatkowo porównane z symulacjami przeprowadzonymi za pomocą komputera PC z zastosowaniem standardowej funkcji RAND. Otrzymane zestawienie wykazało, że generator z 64-bitowym rejestrem LFSR jest równie skuteczny co pozostałe rozwiązania. Wymaga on przy tym niemalże dwukrotnie mniejszej ilości zasobów sprzętowych niż jego 128-bitowy odpowiednik. Biorąc dodatkowo pod uwagę fakt, że każda komórka KDLL jest wyposażona w swój własny generator, zastosowanie w finalnej wersji generatora 64-bitowego pozwoliło zmniejszyć liczbę modułów logicznych potrzebnych dla uzyskania otrzymanej implementacji.

Innym testem było sprawdzenie poprawności i wydajności realizacji algorytmu DLL. Ze względu na brak odniesienia do innej, podobnej do mDLL maszyny, zdecydowano się na porównanie szybkości pomiędzy realizacją określonej symulacji za pomocą mDLL i komputera PC. Do tego celu w komputerze uruchamiano specjalnie przygotowaną aplikację pozwalającą na emulację pracy mDLL. Zastosowano komputer posiadający 2 GB pamięci operacyjnej i dwurdzeniowy procesor Intel Pentium D 3.4 GHz, przy czym referencyjna aplikacja wykorzystywała tylko jeden z rdzeni procesora. Analiza wydajności obliczeniowej urządzenia mDLL wykazała, że jego przewaga nad typowym komputerem osobistym jest co najmniej 40-krotna.

Ze względu na szeregową wymianę danych między modułami, wykonanie jednego kroku symulacji dla jednej cząsteczki cieczy wymagało średnio 2-krotnie większej liczby cykli zegara systemowego niż w przypadku referencyjnej implementacji sekwencyjnej uruchomionej na komputerze klasy PC. Częstotliwość zegara systemowego w wykonanym prototypie wynosiła zaledwie 100 MHz, czyli była 34 razy mniejsza niż w użytych do testów procesorze, co w połączeniu z 2-krotnie większą liczbą cykli dało niemalże 70-krotnie dłuższy czas wykonania jednego kroku symulacji dla jednej cząsteczki. Należy jednak zaznaczyć, że w przeciwieństwie do standardowego procesora, w mDLL można było wykonać jednoczesne symulacje dla układu zawierającego ponad 3 000 cząsteczek.

Równie dobrze wyglądało porównanie poboru mocy. Założenie, że typowy komputer osobisty potrzebuje 200 W by zasilić płytę główną, procesor i pamięć operacyjną, w konfrontacji z faktem, że maksymalny pobór mocy wykonanego prototypu to 1000 W, doprowadziło do konkluzji, że 40-krotne przyspieszenie obliczeń okupione było w przypadku mDLL jedynie 5-krotnym wzrostem zapotrzebowania na energię elektryczną.



Rys. 9 Rozrzut liczby odwiedzin węzłów w funkcji liczby kroków dla różnych metod generacji liczb losowych

Podsumowanie

Zaprojektowanie, wykonanie i testowanie maszyny mDLL doprowadziło do opracowania takiej jej konstrukcji, w której komórki operacyjne (np. KDLL) znajdowały się w węzłach sieci trójwymiarowego torusa i jednocześnie urządzenie było skalowalne. Oznaczało to, że przyszła rozbudowa tego urządzenia o dodatkowe płyty PCB nie będzie powodowała wydłużenia połączeń przewodowych pomiędzy układami FPGA i nie spowolni działania maszyny. Przeprowadzone testy potwierdziły poprawność przyjętych założeń konstrukcyjnych oraz pokazały, że z użyciem mDLL można efektywnie wykonywać symulacje molekularne.

Model maszyny mDLL posiadał też wady, które nie wpływały na jej działanie, ale należałoby je wyeliminować budując następne urządzenie. Jedną z nich była konieczność naprzemiennego mocowania paneli, z których jeden zawierał *plaszczynę prostą*, a drugi *plaszczynę obróconą*. Pomimo, że sumaryczna liczba układów *podstawowych* i *uzupełniających* na dwóch sąsiednich płytach PCB, na sąsiadujących ze sobą panelach była taka sama (4 układy *podstawowe* i 4 *uzupełniające*), to jednak, zależnie od parzystości liczby węzłów (α , β i η) implementowanych w układzie FPGA, na każdej z płyt PCB liczba tych układów mogła być inna (np. na jednej płycie znajdowały się 4 układy *podstawowe*, a na drugiej 4 *uzupełniające*). Oznaczało to konieczność opracowania ośmiu różnych podzbiorów plików konfiguracyjnych, a rozbudowa maszyny wymagała zamontowania co najmniej dwóch dodatkowych płyt PCB, po jednej na sąsiadujących ze sobą panelach. Dodatkowo przewody przeznaczone do transmisji sygnałów pomiędzy komórkami znajdującymi się na różnych panelach, w kłopotliwy sposób, musiały być „przeplatane” co drugi panel.

Pomimo pewnych niedociągnięć konstrukcyjnych maszyna mDLL stanowiła prototyp, który był już w wystarczającym stopniu przetestowany do tego, aby zastosowane w nim rozwiązania technologiczne można było użyć do budowy urządzenia o liczbie zawierającej od 1 mln do 5 mln komórek KDLL. Takie urządzenie nadawałoby się już do prowadzenia symulacji układów wielocząsteczkowych z niespotykaną dotąd szybkością.

Podziękowanie

Projekt zrealizowano w ramach grantu MNiSW, N515 336136 Nr 3361/B/T02/2009/36.

Autorzy: dr hab. inż. Jarosław Jung, Katedra Fizyki Molekularnej ul. Żeromskiego 116, 90-924 Łódź, E-mail: jaroslaw@p.lodz.pl; dr inż. Rafał Kiełbik, Politechnika Łódzka, Katedra Mikroelektroniki i Technik Informatycznych ul. Żeromskiego 116, 90-924 Łódź, E-mail: rkiełbik@dmcs.p.lodz.pl;

dr inż. Kamil Rudnicki, Brightelligence Ltd 2/3, 1 Garriochmill Way G20 6LL, Galsgow, UK, E-mail: rudnicki.dll@gmail.com;

dr hab. Piotr Polanowski Politechnika Łódzka, Katedra Fizyki Molekularnej ul. Żeromskiego 116, 90-924 Łódź, E-mail: ppolanow@p.lodz.pl

LITERATURA

- [1] Fitch B. G., Rayshubsky A. , Elftheriou M., Ward T. J. C., Giampapa M. E., Pitman M. C., Pitera J.W., Swope W.C., Germain R. S , Blue Matter: Scalling of Nbody simulations to one atom per node, *IBM J. Res & Dev*, 52 (2008), 145
- [2] Hu, Xiaohu, Hong L., Smith M. D., Neusius T., Cheng X., Smith J. C., The Dynamics of Single Protein Molecules Is Non-Equilibrium and Self-Similar over Thirteen Decades in Time, *Nature Physics* 12(2016), no. 2, 171–174.
- [3] Chen W., De Schutter E., Parallel STEPS: Large Scale Stochastic Spatial Reaction-Diffusion Simulation with High Performance Computers, 11(2017), *Frontiers in Neuroinformatics*, doi: 10.3389/fninf.2017.00013
- [4] Polanowski P., Implementacja sieciowych modeli cieczy i polimerów w symulacjach opartych na obliczeniach równoległych, *Rozprawa doktorska*, (2002), Politechnika Łódzka
- [5] Jackson R. D., Hoane A. J. Jr., Modular Infinitely Extentable Three Dimensional Torus Packing Scheme for Parallel Processing, (1998), *US Patent no. US005715391A*, <http://www.google.ch/patents/US5715391>
- [6] Pakula T., Collective dynamics in simple supercooled and polymer liquids, *Journal of Molecular Liquids*, 86(2000), 109-121
- [7] Polanowski P., Jeszka J. K., Krysiak K., Matyjaszewski K., Influence of intramolecular crosslinking on gelation in living copolymerization of monomer and divinyl cross-linker. Monte Carlo simulation studies, *Polymer*, 79(2015), 171-178.
- [8] Polanowski P., Jung J., Kiełbik R., Napieralski A., Lichy K., Od algorytmu dynamicznej cieczy sieciowej do dedykowanego komputera równoległego, *Przegląd Elektrotechniczny*, 84 (2008), n.11, 69-73
- [9] Jung J., Od wybranych zagadnień elektroniki organicznej do uniwersalnej maszyny przeznaczonej do analizy zjawisk zachodzących w gęstych układach wieloskładnikowych, *Rozprawa habilitacyjna*, (2016), Politechnika Łódzka
- [10] Jung J., Polanowski P., Kiełbik R., Special state machine based on Dynamic Lattice Liquid model, *International Journal of Engineering Science and Innovative Technology (IJESIT)*, 3(2014), 360-367
- [11] Jung J., Polanowski P., Kiełbik R., Hałagan K., Zatorski W., Ulański J., Napieralski A., Pakula T., A parallel machine with reduced number of connections between logical circuits, (2016), *EP Patent Application no. EP3079073A1*, 2016, <https://www.google.com/patents/EP3079073A1?cl=en>
- [12] Pachanek G.G., Pitsianis N. P., Barry E. F., Drabenstott T. L., Method and Apparatus for Manifold Array Processing, (2000), *US Patent no. US6167502A*, <https://www.google.tl/patents/US6167502>
- [13] Daniel H. W., Parallel Processor, (1992), *EP Patent no. EP0501524A2*, <https://www.google.com.na/patents/EP0501524A2?cl=en>
- [14] Passint R. S., Thorson G., Galles M. B., Hybrid Hypercube/Torus Architecture, (2001), *US Patent no. US6230252B1*, <http://www.google.ch/patents/US6230252>
- [15] Jung J., Polanowski P., Kiełbik R., Hałagan K., Zatorski W., Ulański J., Napieralski A., Pakula T., Panel z układami elektronicznymi i zestaw paneli, (2016), *RP Patent no. P.405479* <http://regserv.uprp.pl/register/application?number=P.405479>
- [16] Jung J., Polanowski P., Kiełbik R., Hałagan K., Zatorski W., Ulański J., Napieralski A., Pakula T., A panel with electronic circuits and a set of panels, (2016), *EP Patent Application no. EP3079071A1* <https://google.com/patents/EP3079071A1?cl=en>
- [17] Jung J., Polanowski P., Kiełbik R., Hałagan K., Zatorski W., Ulański J., Napieralski A., Pakula T., A parallel machine having operational cells located at nodes of a face centered lattice, (2016), *EP Patent Application no. EP3079072A1*, <https://www.google.com.na/patents/EP3079072A1?cl=en>