POZNAN UNIVERSITY OF TECHNOLOGY ACADEMIC JOURNALS
No 85 Electrical Engineering 2016

Wojciech PIETROWSKI*
Grzegorz D. WIŚNIEWSKI
Konrad GÓRNY

# WIDMOWA I FALKOWA ANALIZA PRĄDU SILNIKA LSPMSM Z WYKORZYSTANIEM OPENCL

W artykule przedstawiono zastosowanie algorytmów obliczeń równoległych oraz funkcji zawartych w bibliotece OpenCL do analizy harmonicznej i analizy falkowej prądu fazowego silnika LSPMSM. Opisano interface programowania OpenCL oraz opracowane oprogramowanie w języku C++, w którym zaimplementowano zarówno algorytmy sekwencyjne realizowane przez CPU jak również algorytmy równoległe realizowane przez GPU. Przedstawiono porównanie czasu obliczeń algorytmem sekwencyjnym oraz algorytmem równoległym.

SŁOWA KLUCZOWE: analiza widmowa, analiza falkowa, silnik LSPMSM, OpenCL, obliczenia równoległe, obliczenia sekwencyjne

## 1. WPROWADZENIE

Rozwój technologii pozwolił na obserwację sygnałów i zapisywanie tych obserwacji do zbioru danych. Ze względu na łatwość w przetwarzaniu, najczęściej stosowane są sygnały elektryczne. Sygnały innego pochodzenia można w łatwy sposób przetworzyć na wielkości elektryczne, co pozwala na dalsze przetwarzanie niesionej informacji. Jednak duża ilość informacji w przetwarzanych sygnałach może w znacznym stopniu utrudnić ich szybką analizę zwłaszcza w układach pracujących w czasie rzeczywistym. Wraz z dynamicznym rozwojem technologii w wielu obszarach coraz większe znaczenie zyskuje zastosowanie sygnałów w postaci cyfrowej a rosnąca ilość przetwarzanych informacji determinuje konieczność stosowania coraz bardziej skutecznych metod ich przetwarzania. Jednym ze sposobów poprawy wydajności oprogramowania jest zastosowanie algorytmów obliczeń równoległych. W ostatnich latach można zaobserwować rozwój metod obliczeń równoległych wykorzystujących systemy heterogeniczne, w których zaimplementowane są funkcje zawarte w bibliotece OpenCL. Obszary zastosowania algorytmów obliczeń równoległych są bardzo duże a należą do nich np. przetwarzanie sygnałów audio i wideo, biologia obliczeniowa i chemia, symulacja dynamiki płynów, diagnostyka urządzeń.

* Politechnika Poznańska.

## 2. INTERFACE PROGRAMOWANIA OPENCL

OpenCL (ang. Open Computing Language) jest zbiorem funkcji i procedur służących do programowania algorytmów obliczeń równoległych na systemach komputerowych wyposażonych w różne rodzaje procesorów, kart graficznych i innych układów specjalizowanych. Standard OpenCL definiuje interfejs programistyczny aplikacji używany na komputerach oraz język programowania C99. W opracowanie standardu OpenCL była na początku zaangażowana tylko firma Apple. Obecnie nad rozwojem pracuje konsorcjum Khronos Group. W standardzie OpenCL przyjęto założenie, że system komputerowy złożony jest z hosta, na którym obecny jest kompilator języka C/C++ oraz z urządzeń obliczeniowych, np. procesor, koprocesor, karta graficzna. Te urządzenia wykonują kernele, tj. kod programu specjalnie przygotowany do współpracy z platformą OpenCL. Przy użyciu jednolitego interfejsu programistycznego można za pomocą architektury OpenCL programować poszczególne elementy systemu hybrydowego, w którego skład wchodzą wielordzeniowe procesory CPU, urządzenia GPU różnych producentów. Model platformy składa się z komputera, który jest hostem oraz dodatkowo urządzenia OpenCL. Aplikacja zostaje uruchomiona na hoście, który zleca wykonywanie obliczeń na urządzeniu równoległego przetwarzania. Program posiada dwie przestrzenie pamięci:

– prywatna, która jest dostępna tylko dla jednego wątku roboczego,
– lokalna, w ramach której wszystkie wątki z bloku roboczego posiadają prawa do zapisu i odczytu, a zdefiniowane zmienne są widoczne tylko dla tych wątków.

Wykonywanie kernela na GPU jest kolejkowane i kontrolowane poprzez hosta. Każda instancja głównej funkcji przetwarzającej jest nazywana wątkiem roboczym. Wykonując te same instrukcje korzysta z innych danych wejściowych. Przed rozpoczęciem wykonywania obliczeń na procesorze graficznym trzeba zdefiniować liczbę wątków roboczych, które będą niezbędne do wykonania przetwarzania wszystkich danych wejściowych. Nazywa się to przestrzenią indeksów. Interfejs OpenCL może obsłużyć maksymalnie 3-wymiarową przestrzeń. W perspektywie ograniczenia wątków roboczych, tj. w celu osiągnięcia najlepszej wydajności zalecane jest użycie ich minimalnej ilości, wynikającej z wymagań algorytmu. Wątki uruchamiane na GPU zarządzane są sprzętowo, więc ich użycie i zarządzanie nimi jest bardzo łatwe w odróżnieniu do CPU. Wątki robocze zebrane są w grupy robocze. Możliwe jest synchronizowanie wątków w grupie roboczej, ale pomiędzy grupami już nie ma takiej możliwości. Każdy wątek roboczy ma swój globalny identyfikator, który pozwala na jego lokalizację w przestrzeni indeksów. Tak samo każda grupa robocza ma swoje unikalne identyfikatory [1].

## 3. ZASTOSOWANIE OPENCL W ANALIZIE SYGNAŁÓW

W artykule skoncentrowano się na przedstawieniu zastosowania algorytmów obliczeń równoległych w analizie harmonicznej oraz analizie falkowej. Opracowane oprogramowanie umożliwia wykonanie obliczeń algorytmem sekwencyjnym przez CPU oraz algorytmem równoległym przez GPU. Obliczenia równoległe realizowane przez GPU były realizowane poprzez zaimplementowanie funkcji zawartych w bibliotece OpenCL. Kod aplikacji obliczeń równoległych dzielony jest na część CPU (hosta) i część GPU (urządzenia na platformie). Systemowy kompilator zajmuje się częścią CPU, natomiast część GPU (kernel) jest tłumaczony na kod maszyny wirtualnej, a następnie do kodu binarnego odpowiedniego dla konkretnego rodzaju urządzenia. Program napisano w środowisku Microsoft Visual Studio. Zastosowano język C++ ze względu na dostępność plików nagłówkowych biblioteki OpenCL. Do testowania napisanego programu wykorzystano zarejestrowane przebiegi prądów fazowych silnika LSPMSM [2].

### 3.1. Analiza widmowa prądu fazowego

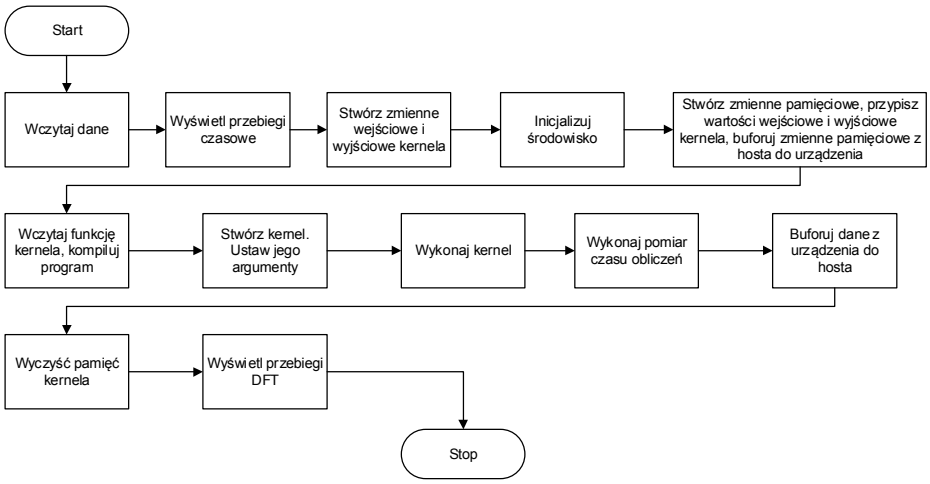
Analiza widmowa prądu jest powszechnie stosową metodą w diagnostyce i monitorowaniu maszyn elektrycznych [W3]. W opracowanym oprogramowaniu do analizy zarejestrowanego przebiegu prądu wykorzystano dyskretną transformatę Fouriera postaci

$$\mathrm{x(m)} = \sum_{n=0}^{N-1} x(n)\cos\left(2\pi\frac{n\cdot m}{N}\right) - j\sum_{n=0}^{N-1} x(n)\sin\left(2\pi\frac{n\cdot m}{N}\right) \tag{1}$$

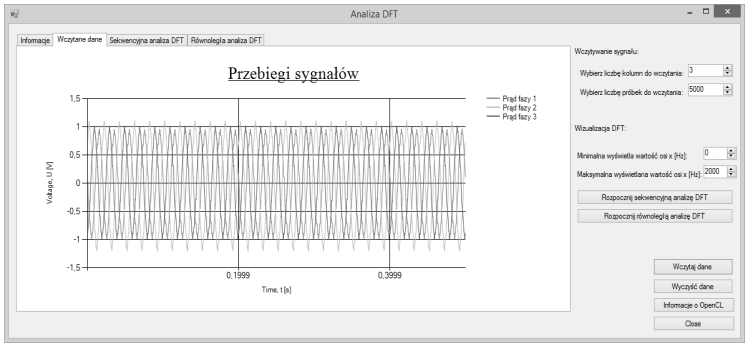
gdzie: $x(n)$ jest sygnałem dyskretnym, $n$ jest kolejną próbką, $m$ jest indeksem próbki na wyjściu, $N$ jest całkowitą liczbą próbek sygnału wejściowego [3].

Algorytm opracowanego programu przedstawiono w postaci schematu blokowego na rys. 1. W programie można wyróżnić następujące główne funkcje:
- wczytanie danych z pliku,
- przetworzenie danych wg wybranego algorytmu,
- wizualizacja wczytanych danych w dziedzinie czasu,
- analizę częstotliwościową DFT zarówno przez CPU jak i przez GPU,
- wizualizacja widma w dziedzinie częstotliwości,
- porównanie czasu obliczeń.

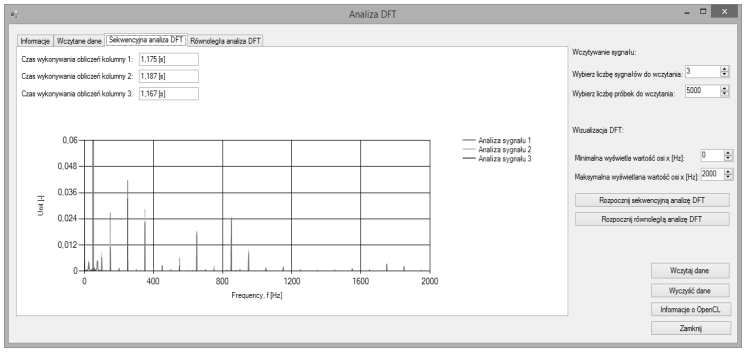
Przykład wizualizacji wczytanych danych w dziedzinie czasu przedstawiono na rys. 2. Przedstawione przebiegi są prądami fazowymi silnika LSPMSM przy obciążeniu silnika stałym w czasie momentem o wartości znamionowej. Dokonano analizy harmonicznej wczytanych przebiegów algorytmem sekwencyjnym i równoległym. Zastosowana analiza widmowa może być przeprowadzona na sygnałach stacjonarnych, dlatego do analizy wybrano przebieg prądu w stanie ustalonym po załączeniu napięcia zasilania. Wyniki obliczeń przedstawiono odpowiednio na rys. 3 i rys. 4.



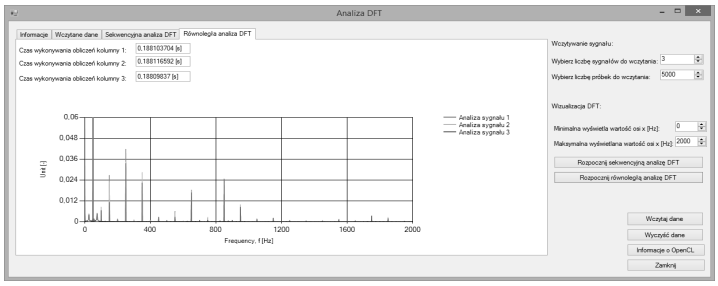

Rys. 1. Algorytm programu realizującego dyskretne przekształcenie Fouriera z wykorzystaniem OpenCL




Rys. 2. Przebiegi prądów fazowych silnika LSPMSM




Rys. 3. Analiza widmowa prądów algorytmem sekwencyjnym




Rys. 4. Analiza widmowa prądów algorytmem równoległym

W celu porównania skuteczności algorytmów obliczeniowych przeprowadzono obliczenia testowe. Jako miarę skuteczności testu przyjęto czas potrzebny na przeprowadzenie analizy sygnału o takiej samej liczbie próbek przez każdy z algorytmów. Czasy obliczeń oznaczono następująco: $t_{CPU}$ – czas obliczeń algorytmem sekwencyjnym realizowanym przez CPU, $t_{GPU}$ – czas obliczeń algorytmem równoległym realizowanym przez GPU. W testach wykorzystano dwa zestawy komputerowe:

Zestaw I

CPU: procesor Intel Core i5 4670k 3,4 GHz,

GPU: układ graficzny Radeon R9 270X o 1280 procesorach strumieniowych.

Zestaw II

CPU: procesor Intel Core 2 Duo E8400 3 GHz,

GPU: układ graficzny Radeon HD5700 o 720 procesorach strumieniowych.

Każdy z zestawów komputerowych wykonał dwa testy obliczeniowe. Test I polegał na pomiarze czasu obliczeń analizy harmonicznej sygnału złożonego z 7 500 próbek. Wyniki testu I przestawiono w tabeli 1. Natomiast test II dotyczył pomiaru czasu dla sygnału składającego z 5 000 próbek. Wyniki testu II przestawiono w tabeli 2.

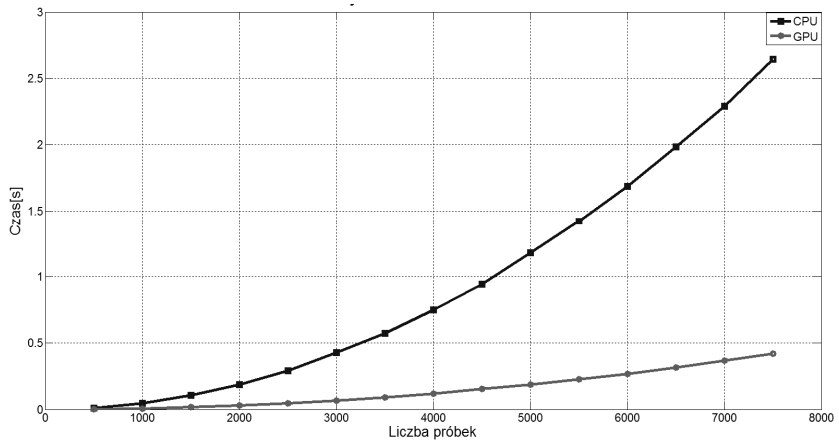
W kolejnych obliczeniach testowych każdy z zestawów komputerowych wykonał obliczenia dla sygnału wejściowego o liczbie próbek w zakresie od 500 do 7500. Wyniki obliczeń przedstawiono na rysunkach 5 i 6.

Tabela 1. Wyniki testu I wykonanego na zbiorze 7 500 próbek

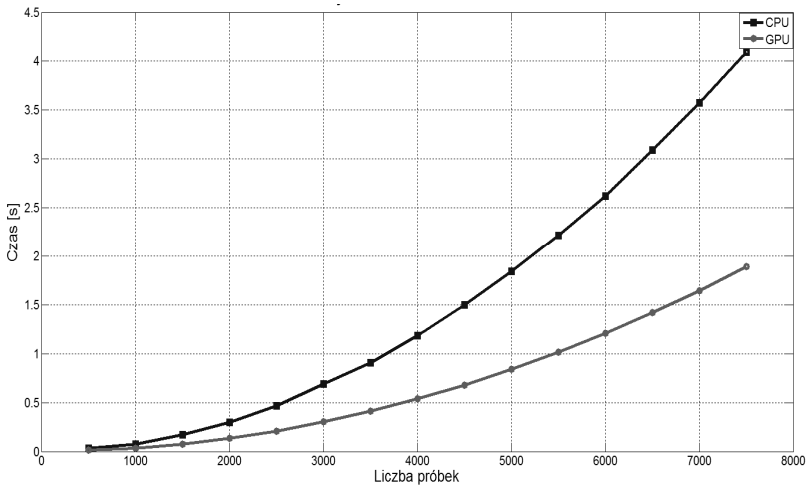
| Sygnał | Zestaw I | | | Zestaw II | | |
|---|---|---|---|---|---|---|
| | $t_{CPU}$ | $t_{GPU}$ | $t_{CPU}/t_{GPU}$ | $t_{CPU}$ | $t_{GPU}$ | $t_{CPU}/t_{GPU}$ |
| | [s] | [s] | [-] | [s] | [s] | [-] |
| Prąd fazy L1 | 2,65 | 0,42 | 6,3 | 4,07 | 1,09 | 3,7 |
| Prąd fazy L2 | 2,62 | 0,42 | 6,2 | 4,04 | 1,09 | 3,7 |
| Prąd fazy L3 | 2,63 | 0,42 | 6,2 | 4,06 | 1,08 | 3,7 |

Tabela 2. Wyniki testu II wykonanego na zbiorze 5 000 próbek

| Sygnał | Zestaw I | | | Zestaw II | | |
|---|---|---|---|---|---|---|
| | $t_{CPU}$ | $t_{GPU}$ | $t_{CPU}/t_{GPU}$ | $t_{CPU}$ | $t_{GPU}$ | $t_{CPU}/t_{GPU}$ |
| | [s] | [s] | [-] | [s] | [s] | [-] |
| Prąd fazy L1 | 1,23 | 0,18 | 6,8 | 1,84 | 0,85 | 2,1 |
| Prąd fazy L2 | 1.21 | 0,18 | 6,7 | 1,84 | 0,84 | 2,1 |
| Prąd fazy L3 | 1,25 | 0,18 | 6,9 | 1,84 | 0,84 | 2,1 |




Rys. 5. Czas obliczeń w funkcji liczby próbek wykonanych przez zestaw I




Rys. 6. Czas obliczeń w funkcji liczby próbek wykonanych przez zestaw II

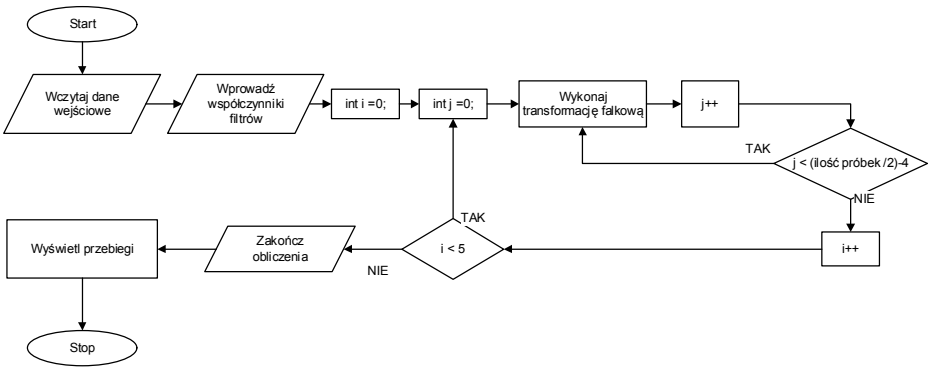
### 3.2. Analiza falkowa prądu fazowego

Jedną z metod analizy sygnałów niestacjonarnych jest metoda wykorzystująca przekształcenie falkowe [4, 5]. W opracowanym oprogramowaniu do analizy sygnałów niestacjonarnych zaimplementowano dyskretną transformatę falkową postaci

$$DWTx_{j,n} = \sum_{n=0}^{N-1} x(n)\psi_{j,n}^{*}(n) \tag{2}$$

Realizacja dyskretnej analizy falkowej polega na dekompozycji sygnału za pomocą pary filtrów dolno i górnoprzepustowego. W wyniku filtracji otrzymuje się aproksymację oraz detal sygnału wejściowego. Każdy z otrzymanych składników można poddać dalszej dekompozycji otrzymując kolejne poziomy dekompozycji sygnału wejściowego. W wyniku filtracji sygnału o n próbkach otrzymuje się sygnał o długości 2n, dlatego stosuje się tak zwany downsampling, czyli usunięcie co drugiej próbki sygnału wyjściowego. Dekompozycję można zrealizować do poziomu dekompozycji zależnego od długości sygnału wejściowego. Każdy z n+1 sygnałów, które są otrzymywane w wyniku dekompozycji oryginalnego sygnału na n-tym poziomie, zawiera składowe oryginalnego sygnału należące do określonego pasma częstotliwości.

Omówiony schemat zaimplementowano w oprogramowaniu własnym napisanym w języku C++. Algorytm programu przedstawiono na rysunku 7.




Rys. 7. Algorytm transformaty falkowej

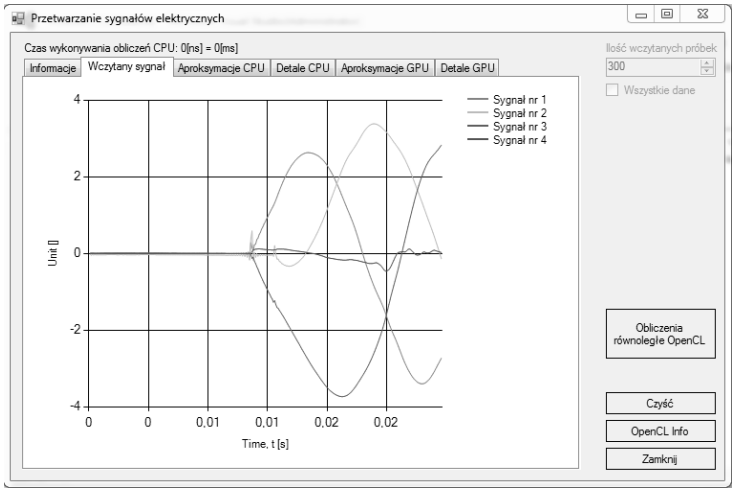
Opracowany program zastosowano do analizy falkowej przebiegów prądu fazowego silnika LSPMSM w chwili załączenia napięcia zasilania. Analizowane przebiegi przedstawiono na rysunku 8. W obliczeniach testowych wykorzystano ten sam zbiór sygnałów jak w przykładzie dotyczącym analizy harmonicznej. W programie zaimplementowano analizę falkową z zastosowaniem falki Daubechies db4, dla której wartości parametrów filtra górnoprzepustowego są następujące:

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}} \qquad h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}} \qquad h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}} \qquad h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$$

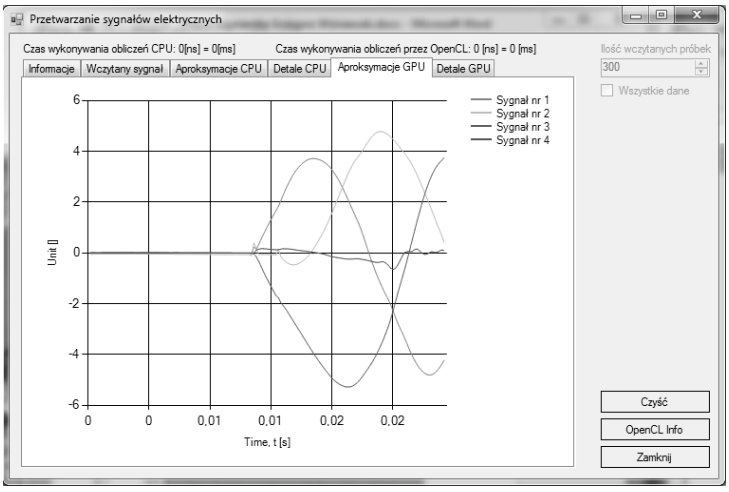
Natomiast filtra dolnoprzepustowego są następujące:

$$g_0 = \frac{1-\sqrt{3}}{4\sqrt{2}} \qquad g_1 = -\frac{3-\sqrt{3}}{4\sqrt{2}} \qquad g_2 = \frac{3+\sqrt{3}}{4\sqrt{2}} \qquad g_3 = -\frac{1+\sqrt{3}}{4\sqrt{2}}$$

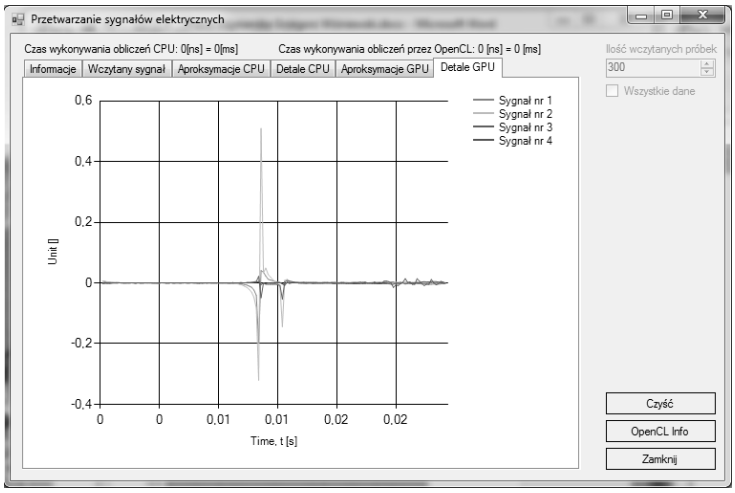
W wyniku dekompozycji analizowanego sygnału otrzymano przebieg aproksymacji i detalu, które przedstawiono na rysunkach 9 i 10.




Rys. 8. Przebieg prądów fazowych w chwili załączenia napięcia zasilania




Rys. 9. Aproksymacja przebiegów prądów po załączeniu napięcia zasilania




Rys. 10. Detal przebiegów prądów po załączeniu napięcia zasilania

## 4. PODSUMOWANIE

W artykule przedstawiono porównanie czasu obliczeń algorytmów równoległych oraz sekwencyjnych w analizie widmowej oraz falkowej prądu silnika LSPMSM. Obliczenia testowe przeprowadzono na dwóch różnych zestawach komputerowych, dla sygnałów o różnej liczbie próbek. Na podstawie otrzymanych wyników przeprowadzonych obliczeń testowych analizy harmonicznej można zaobserwować kilkukrotne zmniejszenie czasu przetwarzania sygnału algorytmem równoległym w porównaniu do obliczeń algorytmem sekwencyjnym. Przewaga algorytmu równoległego jest tym większa im więcej próbek sygnału jest przetwarzana.

## LITERATURA

[1] Sawerwain M., OpenCL Akceleracja GPU w praktyce, PWN 2014.
[2] Idziak P., Nowak M., Pietrowski W., Spectral analysis of phase currents of LSPMSM at asymmetric voltage supply, Pomiary Automatyka Kontrola nr 10, 2013, pp. 1032-1035.
[3] Szabatin J., Podstawy teorii sygnałów, Wydawnictwo WKiŁ, 2007.
[4] Kowalski Cz. T., Diagnostyka układów napędowych z silnikiem indukcyjnym z zastosowaniem metod sztucznej inteligencji, Oficyna Wydawnicza Politechniki Wrocławskiej, 2013.
[5] Dziechciarz A., Sułowicz M., Zastosowanie analizy falkowej do diagnozowania uszkodzeń w silniku indukcyjnym podczas pracy przy zmiennym obciążeniu, Elektrotechnika Czasopismo Techniczne, Wydawnictwo Politechniki Krakowskiej, Kraków 2009.

## SPECTRAL AND WAVELET ANALYSIS OF PHASE CURRENT OF LSPMSM MACHINE USING OPENCL

The article presents a comparison of a computing time of a parallel and a sequential algorithm in a spectral and a wavelet analysis of a motor LSPMSM current. The test calculations were made on two different sets of computer for different number of signals samples. On the basis of the results of test calculations of harmonic analysis it can be observed that using parallel algorithm a signal processing time has been reduced of several times compared to a sequential algorithm. The advantage of the parallel algorithm is the greater, the more signal samples are processed.

*(Received: 18. 02. 2016, revised: 7. 03. 2016)*