Równoległy algorytm analizy sygnału na podstawie niewielkiej liczby próbek

Piotr Kardasz

Wydział Elektryczny, Politechnika Białostocka

Streszczenie: W artykule przedstawiono równoległy algorytm estymacji parametrów składowych sinusoidalnych złożonego sygnału. Proponowany algorytm umożliwia rozpoznanie składowych sygnału również w warunkach, gdy dysponujemy ograniczoną liczbą losowo pobranych próbek tego sygnału. Zbadany został czas pracy zaproponowanego algorytmu w funkcji liczby równocześnie uruchomionych wątków. Do testowania zostały zastosowane komputery o różnej liczbie rdzeni procesora, obsługiwanych wątków oraz zmiennoprzecinkowych jednostek wykonawczych. Wyniki eksperymentu pokazują, że proponowany algorytm może pracować efektywnie, nawet jeśli liczba wątków obliczeniowych przekracza liczbę jednostek wykonawczych procesora, na którym pracuje. W artykule zostały również zarysowane kierunki dalszych badań nad udoskonaleniem przedstawionego algorytmu.

Słowa kluczowe: algorytm równoległy, identyfikacja sygnału

DOI: 10.14313/PAR_204/112

1. Wprowadzenie

Klasyczne algorytmy analizy i przetwarzania sygnałów jednowymiarowych przedstawionych w postaci dyskretnej, oparte są na wykorzystaniu jednej z popularnych transformat, takich jak transformata Fouriera, kosinusowa lub falkowa [1]. Metody te mogą być stosowane tylko wtedy, gdy jest dostępny pełny zestaw próbek analizowanego sygnału, pobranych z częstotliwością próbkowania równą lub wyższą niż wynikająca z twierdzenia Kotielnikowa-Shannona. Warunek ten nie zawsze może być spełniony. Na skutek uszkodzeń nośnika, na którym zapisany jest sygnał, zakłóceń zewnętrznych bądź przerw w torze transmisyjnym możemy dysponować tylko częścią próbek, niezbędnych do analizy badanego sygnału.

Istnieją również przypadki, w których pobranie pełnej liczby próbek badanego sygnału jest ze względów technicznych niemożliwe, bądź też zbyt kosztowne [4]. Można wtedy zastosować intensywnie rozwijającą się w ostatnich latach metodę analizy i transmisji sygnału, określaną jako *Compressed Sensing* [2, 3]. Polega ona na przygotowaniu niewielkiej liczby próbek sygnału i rekonstrukcji tego sygnału na ich podstawie. Jedna z najprostszych metod tego rodzaju polega na losowym odrzuceniu części próbek [2] i rekonstrukcji sygnału na podstawie pozostałych.

Rekonstrukcja sygnału na podstawie tego rodzaju danych jest możliwa za pomocą różnego rodzaju algorytmów. Stosowane są w tym celu metody typu *basis pursuit* [3]. Możliwe są również inne podejścia, takie jak algorytmy ewolucyjne [5–7, 10] i inne metody poszukiwania składowych rekonstruowanego sygnału [11]. Wszystkie te algorytmy charakteryzują się jednak wysoką złożonością obliczeniową, a co za tym idzie, narzucają wysokie wymagania na moc obliczeniową sprzętu, na którym są wykonywane. Jeśli jest ona zbyt niska, obliczenia wykonują się zbyt wolno, w szczególności nie mogą być wykonywane w czasie rzeczywistym.

Klasyczne procedury obliczeniowe są wykonywane szeregowo, instrukcja po instrukcji, w jednym wątku. Współczesny sprzęt komputerowy jest jednak wyposażony w wielordzeniowe mikroprocesory. Każdy z tych rdzeni jest ma kilka jednostek arytmetyczno-logicznych, mogących wykonywać operacje stałoprzecinkowe lub zmiennoprzecinkowe. Umożliwia to równoległą pracę wielu wątków obliczeniowych na tym samym procesorze. Algorytmy, które mogą wykorzystać możliwości pracy wielowątkowej, mogą dzięki temu pracować szybciej, w pełni wykorzystując możliwości CPU. Procesory graficzne (GPU) dysponują jeszcze większą (nawet ponad 2000) liczbą zmiennoprzecinkowych jednostek obliczeniowych, które mogą zostać użyte do obliczeń dzięki zastosowaniu bibliotek takich jak OpenCL, CUDA lub DirectCompute [8]. Wykorzystanie możliwości równoległego wykonywania obliczeń mogłoby pozwolić na znaczne przyspieszenie pracy algorytmów analizy sygnałów.

Innym sposobem uzyskania wyników obliczeń w krótkim czasie jest zastosowanie programowalnych układów logicznych (FPGA) [9], które poza tym, że stwarzają możliwość zaimplementowania równoległych algorytmów obliczeniowych, pozwalają także na dostosowanie możliwości projektowanych jednostek obliczeniowych do konkretnych potrzeb danego algorytmu.

W większości przypadków algorytmów równoległych nie jest możliwe równoległe wykonanie całej pracy. Część zadań takiego algorytmu, na przykład przygotowanie danych i zebranie końcowych wyników obliczeń, musi być wykonana w pojedynczym wątku. Nie jest również na ogół możliwe doskonale równomierne rozdzielenie obli-

czeń pomiędzy wątki, wskutek czego o czasie pracy całego algorytmu będzie decydował ten wątek, który najpóźniej zakończy obliczenia. W tej sytuacji zwiększanie liczby wątków nie skraca w sposób proporcjonalny czasu obliczeń. Zakładając, że mamy do dyspozycji nieograniczoną liczbę jednostek wykonawczych, wraz ze wzrostem liczby wątków czas ten będzie dążył asymptotycznie do pewnej niezerowej minimalnej wartości, wyznaczonej przez te elementy algorytmu, które muszą zostać wykonane w jednym wątku, oraz minimalny czas obliczeń dla pojedynczego wątku. Ponieważ rzeczywiste procesory dysponują skończoną liczbą jednostek wykonawczych, zwiększenie liczby wątków powyżej liczby tych jednostek nie pozwoli na dalsze przyspieszenie pracy algorytmu, przeciwnie – czas zużyty na przełączanie wątków spowoduje wzrost całkowitego czasu obliczeń.

2. Cel i zakres badań

Jednowątkowe algorytmy [10, 11] rozpoznawania składowych sinusoidalnych sygnału na podstawie niewielkiej liczby jego próbek pracują długo ze względu na ich dużą złożoność obliczeniową. Celem badań było więc opracowanie i przetestowanie, na bazie istniejącego algorytmu jednowątkowego [11], równoległego algorytmu rozpoznawania składowych sinusoidalnych złożonego sygnału na podstawie jego losowo pobranych próbek i określenie, na podstawie wyników pomiarów, czy jest możliwe przeniesienie tego algorytmu do środowisk o dużej liczbie równolegle wykonywanych wątków, takich jak procesory GPU i układy FPGA. Algorytm tego rodzaju powinien pozwolić na uruchomienie go przy zadanej przez użytkownika liczbie wątków.

Algorytm taki został opracowany i uruchomiony. Zbadany został czas wykonania opracowanego algorytmu na komputerach z procesorami o różnej liczbie i strukturze rdzeni w funkcji liczby przetwarzanych równolegle wątków.

Na podstawie przeprowadzonych pomiarów została określona możliwość przeniesienia badanego do środowisk o dużej liczbie jednostek wykonawczych, takich jak procesory GPU i układy FPGA oraz możliwości i kierunki dalszego rozwoju tego algorytmu.

3. Badany algorytm

3.1. Podstawy teoretyczne

Podstawą do opracowania proponowanego algorytmu [11] stało się spostrzeżenie, że iloczyn dwóch funkcji sinusoidalnych, który można przedstawić w postaci

$$\sin(at)\cdot\sin(bt)=\frac{\cos(a-b)t}{2}-\frac{\cos(a+b)t}{2} \tag{1}$$

redukuje się do

$$\sin(at)\cdot\sin(at)=\frac{1}{2}-\frac{\cos(2at)}{2} \tag{2}$$

w przypadku, gdy $a = b$.

Oznacza to, że iloczyn takich funkcji zawiera niezerową składową stałą wtedy i tylko wtedy, gdy częstotliwości obu przebiegów sinusoidalnych są sobie równe; w przeciwnym przypadku składowa ta jest zerowa. Jeśli więc scałkujemy taki sygnał, otrzymamy dla $a \neq b$ funkcję okresową:

$$\int\sin(at)\cdot\sin(bt)dt=\frac{\sin(a-b)t}{2(a-b)}-\frac{\sin(a+b)t}{2(a+b)}+C \tag{3}$$

ale dla $a = b$ funkcja ta będzie niemalejąca:

$$\int\sin(at)\cdot\sin(at)dt=\frac{t}{2}-\frac{\sin(2at)}{2a}+C \tag{4}$$

Jeśli więc sygnał badany ma postać sinusoidalną o amplitudzie A:

$$f(t)=A\sin(\mathrm{at}) \tag{5}$$

zaś sygnał testujący ma tę samą częstotliwość, lecz jednostkową amplitudę, możemy na podstawie iloczynu tych sygnałów określić amplitudę sygnału badanego. Ponieważ:

$$\int_{t_1}^{t_2}A\sin(at)\cdot\sin(at)dt=\frac{A(t_2-t_1)}{2}-\frac{A\sin(2at_2)}{2a}+\frac{A\sin(2at_1)}{2a} \tag{6}$$

więc

$$A=\frac{2\int_{t_1}^{t_2}A\sin(at)\cdot\sin(at)dt}{(t_2-t_1)-\frac{1}{a}\sin(2at_2)+\frac{1}{a}\sin(2at_1)} \tag{7}$$

Jeśli $t_2 - t_1 >> 2/a$, można określić w przybliżeniu

$$A\approx\frac{2\int_{t_1}^{t_2}A\sin(at)\cdot\sin(at)dt}{(t_2-t_1)} \tag{8}$$

a dla sygnałów dyskretnych:

$$A\approx\frac{2\sum_{n=n_1}^{n_2}A\sin(anT)\cdot\sin(anT)}{(n_2-n_1)} \tag{9}$$

Jeśli badany sygnał jest funkcją sinusoidalną przesuniętą w fazie w stosunku do funkcji testującej, możemy rozłożyć go na składową sinusoidalną i kosinusoidalną

$$\sin(at+\varphi)=\sin at\cos\varphi+\cos at\sin\varphi \tag{10}$$

i oszacować amplitudy niezależnie dla obu składowych, po czym korzystając ze wzorów:

$$A = \sqrt{A_{\sin}{}^2 + A_{\cos}{}^2} \tag{11}$$

$$\varphi = arctg\ \frac{A_{\sin}}{A_{\cos}} \tag{12}$$

określić amplitudę i fazę badanego sygnału.

Jeśli z jakiejś przyczyny mamy do dyspozycji tylko część próbek badanego sygnału, równanie (9) przyjmie postać:

$$A \approx \frac{2\sum_{N} P_n \sin(anT)}{N} \tag{13}$$

gdzie N jest liczbą posiadanych próbek, a P_n jest wartością n-tej posiadanej próbki. Możemy więc nadal oszacować amplitudę badanego sygnału, choć z mniejszą dokładnością.

Na podstawie powyższych spostrzeżeń został opracowany, uruchomiony i przetestowany jednowątkowy algorytm [11], umożliwiający identyfikację parametrów składowych sinusoidalnych złożonego sygnału. Algorytm ten był w stanie odnaleźć, z dokładnością rzędu kilku procent, siedem z ośmiu składowych sygnału na podstawie 128 próbek pobranych losowo spośród pełnego zestawu 2048 próbek. Wyniki te są porównywalne z wynikami działania algorytmu ewolucyjnego [10], przy znacznie większej szybkości obliczeń.

3.2. Idea i działanie algorytmu

Czas pracy algorytmu opisanego w [11], mimo że o rząd wielkości krótszy, niż w przypadku algorytmu ewolucyjnego [10], okazał się jednak zbyt długi, aby można go było zastosować w praktyce, np. w celu analizy sygnałów akustycznych. Analiza zestawu 2048 próbek, co dla częstotliwości próbkowania 44 100 Hz stanowi 46,44 ms sygnału, trwała kilkadziesiąt sekund, czyli około tysiąckrotnie dłużej niż czas trwania sygnału.

W tej sytuacji powstała idea opracowania wielowątkowej wersji tego algorytmu. Tak, jak i jednowątkowa jego wersja [11], algorytm dokonuje mnożenia badanego sygnału przez sygnały testowe postaci

$$F_{\sin} = \sin(2\pi ft) \tag{14}$$

oraz

$$F_{\cos} = \cos(2\pi ft) \tag{15}$$

i na podstawie wyników tych mnożeń, zgodnie ze wzorami (11), (12) i (13), oblicza amplitudę i fazę dla każdej z testowanych częstotliwości. Następnie poszukiwane są wartości maksymalne amplitudy w funkcji badanej częstotliwości. Odnalezione maksima traktowane są jako estymaty składowych badanego sygnału. Częstotliwość sygnałów testowych zmienia się z krokiem 1 Hz.

Mnożenie przez każdy z sygnałów testowych może być dokonane niezależnie od pozostałych. Dzięki temu algorytm może być przygotowany w postaci wielowątkowej. W szczególności, gdybyśmy dysponowali sprzętem mogącym obsługiwać tyle wątków, ile jest sygnałów testowych, wszystkie mnożenia mogłyby być wykonywane równolegle, co mogłoby skrócić obliczenia tyle razy, ile jest sygnałów testowych. W przypadku badanego algorytmu mamy do czynienia z 2048 sygnałów testowych, więc w idealnym przypadku można przyspieszyć obliczenia $2 \cdot 10^3$-krotnie – wystarczająco, aby algorytm pracował w czasie rzeczywistym, pod warunkiem, że dysponujemy odpowiednią liczbą jednostek obliczeniowych. Warunek ten spełniają nowoczesne karty graficzne oraz układy wykorzystujące logikę programowalną (FPGA). Opracowanie algorytmów dla tych środowisk jest jednak czasochłonne i wymaga zastosowania odpowiednich narzędzi (języki HDL, środowiska OpenCL lub CUDA [8, 9]). Celowym więc wydawało się – przed przystąpieniem do opracowania programu dla wymienionych środowisk – przetestowanie możliwości algorytmu równoległego w klasycznym środowisku PC. Współczesne procesory klasy x86 pozwalają na równoległą obsługę od 2 do 12 wątków.

W rzeczywistych algorytmach równoległych czas obliczeń jest większy niż wynikałoby to z podzielenia czasu pracy algorytmu jednowątkowego przez liczbę wątków. Dzieje się tak dlatego, że część zadań realizowanych przez algorytm, np. przygotowanie danych i opracowanie wyników musi odbywać się w jednym wątku. Dodatkowo, jeśli wątków jest więcej niż jednostek obliczeniowych maszyny, na której wykonywany jest algorytm, potrzebny jest dodatkowy czas na przełączanie zadań.

W przypadku badanego algorytmu, oprócz procesów przygotowujących dane, nie można wykonać równolegle części obliczeń, polegających na odnalezieniu maksimów wśród wyników mnożeń. Poza tym procedura odnajdująca te maksima musi czekać, aż zakończy się praca ostatniego z wątków wykonujących mnożenia.

Algorytm został opracowany w środowisku uruchomieniowym Lazarus przy użyciu modułu MTProcs. Moduł ten pozwala na łatwą implementację algorytmów równoległych przez obsługę równoległego uruchomienia wielu instancji tej samej procedury. Liczba uruchamianych wątków zadawana jest przez użytkownika; każda z instancji procedury otrzymuje swój niepowtarzalny indeks, dzięki czemu może wykonać swoją część pracy niezależnie od innych instancji. W efekcie przetestowano czas pracy programu w funkcji liczby pracujących równolegle wątków.

Czas pracy mierzony był za pomocą komponentu EpikTimer, korzystającego ze sprzętowych mechanizmów udostępnianych przez procesory klasy x86 – pozwalającego na pomiar czasu z mikrosekundową rozdzielczością. Oprócz czasu przeznaczonego na obliczenia, mierzony był także (w celach poglądowych i porównawczych) czas pracy całego programu oraz czas przeznaczony na wizualizację wyników.

4. Przebieg i wyniki badań

Wstępny etap testów obejmował sprawdzenie, czy wyniki działania badanego algorytmu są równoważne z wynikami algorytmu jednowątkowego [11] oraz sprawdzenie, czy

zestaw danych testowych ma wpływ na czas pracy algorytmu. Badany algorytm okazał się, zgodnie z założeniami, równoważny algorytmowi jednowątkowemu, dając te same wyniki, a postać danych testowych nie miała wpływu na czas pracy algorytmu. W tej sytuacji do dalszych pomiarów parametry sygnału testowego zostały wylosowane przy użyciu generatora liczb pseudolosowych (tab. 1).

Pomiary czasu pracy badanego algorytmu zostały wykonane na komputerach, których parametry zostały przedstawione w tab. 2.

Tab. 1. Parametry sygnału testowego
Tab. 1. Parameters of the test signal

Składowa	Amplituda A	Częstotliwość f [Hz]	Faza φ[stopnie]
1	0,944	1992	92
2	0,471	1006	–101
3	0,350	1358	90
4	0,253	1759	162
5	0,129	1866	–128
6	0,129	1704	–13
7	0,073	503	104
8	0,068	411	126

Tab. 2. Parametry komputerów użytych do testów
Tab. 2. Parameters of the computers used for testing

Lp.	Procesor	Liczba rdzeni	Liczba wątków	Częstotliwość pracy [GHz]
1	AMD FX-8320	8 (4 moduły)	8	4,2
2	AMD FX-8320 (tryb 4 rdzeni)	4	4	4,2
3	Intel i5	4	4	3,3
4	Intel i3	2	4	3,8
5	Intel i7	6	12	3,3
6	AMD A64x2	2	2	2,4

Tab. 3. Czasy obliczeń w funkcji liczby wątków. Żółtym kolorem oznaczono minimalny czas obliczeń dla każdego z zestawów testowych
Tab. 3. Computation time as a function of the number of threads. Yellow indicates the minimum calculation time for each of tested computers

Liczba wątków	Czas obliczeń [s] dla poszczególnych komputerów					
	1	2	3	4	5	6
1	18,34	20,06	19,99	15,34	21,17	21,47
2	9,75	10,76	10,05	7,75	10,63	11,08
3	6,77	6,91	6,78	6,11	7,22	12,29
4	5,46	5,22	5,27	4,71	5,52	11,04
5	5,86	7,82	8,10	5,90	4,70	12,72
6	4,88	7,19	7,36	5,34	4,03	11,59
7	4,70	5,97	6,29	5,36	4,28	11,51
8	4,62	5,42	5,40	4,77	4,25	11,18
9	5,35	6,51	7,20	5,40	3,76	11,51
10	5,15	5,93	6,69	5,11	3,73	11,28
11	4,92	5,91	6,21	5,22	3,80	11,59
12	4,74	5,81	5,62	4,78	3,58	11,40
13	4,70	6,39	7,31	5,09	4,15	11,40
14	4,56	5,93	6,48	5,01	4,06	11,14
15	4,52	5,87	6,06	4.89	3,69	11,44
16	4,59	5,81	5,61	4,80	3,78	11,38
20	4,73	5,80	5,67	4,82	3,35	11,22
24	4,40	5,80	5,67	4,83	3,21	11,34
28	4,53	5,82	5,72	4,81	3,34	11,42
32	4,39	5,84	5,63	4,82	3,33	11,47

Tab. 4. Względna prędkość obliczeń w funkcji liczby wątków
Tab. 4. Relative computation speed as a function of the number of threads

Liczba wątków	Względna prędkość obliczeń (1 wątek = 100 %) dla poszczególnych komputerów [%]					
	1	2	3	4	5	6
1	100,0	100,0	100,0	100,0	100,0	100,0
2	188,1	186,4	198,9	197,9	199,2	193,8
3	270,9	290,3	294,8	251,1	293,2	174,7
4	297,7	384,3	379,3	325,7	383,5	194,5
5	319,5	256,5	246,8	260,0	450,4	168,8
6	375,8	279,0	271,6	287,3	525,3	185,2
7	390,2	336,0	317,8	286,2	494,6	186,5
8	397,0	370,1	370,2	321,6	498,1	192,0
9	342,8	308,1	277,6	284,1	563,0	186,5
10	356,1	338,3	298,8	300,2	567,6	190,3
11	372,8	339,4	312,9	293,9	557,1	185,2
12	386,9	345,3	355,7	320,9	591,3	188,3
13	390,2	313,9	273,5	301,4	510,1	188,3
14	402,2	338,3	308,5	306,2	521,4	192,7
15	405,8	341,7	329,9	313,7	573,7	187,7
16	399,6	345,3	356,3	319,6	560,1	188,7
20	387,7	345,9	352,6	318,3	631,9	191,4
24	416,8	345,9	352,6	317,6	659,5	189,3
28	404,9	344,7	349,5	318,9	633,8	188,0
32	417,8	343,5	355,1	318,3	635,7	187,2

Zestaw testowy nr 1 i 2 to komputer wyposażony w procesor AMD FX-8320 taktowany częstotliwością 4,2 GHz. Procesor ten składa się z czterech modułów, z których każdy zawiera dwie jednostki wykonawcze, widziane przez system operacyjny jako niezależne rdzenie. Jeden rdzeń może obsługiwać jeden wątek. Każdy moduł zawiera jedną jednostkę zmiennoprzecinkową, współużytkowaną przez oba rdzenie. Procesor ten można przełączyć w tryb pracy, w którym każdy modułów pracuje jako pojedynczy rdzeń, co teoretycznie powinno przyspieszyć obliczenia wykonywane przy niewielkiej liczbie wątków.

Zestaw nr 3 zawiera procesor Intel I5-2500K pracujący z częstotliwością 3,3 GHz. Ma on cztery rdzenie, z których każdy obsługuje jeden wątek.

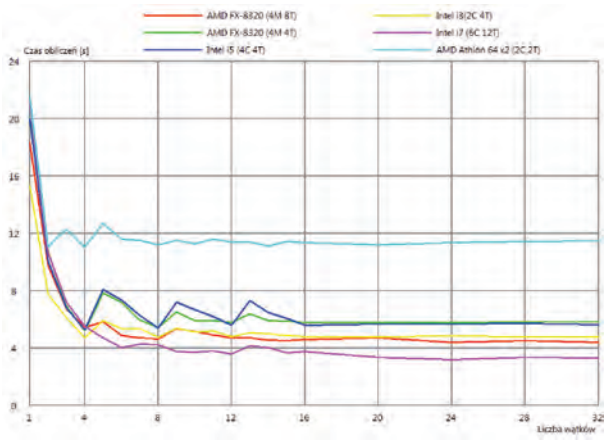
Zestaw nr 4 zawiera procesor Intel i3. Procesor ten ma dwa rdzenie, z których każdy może obsłużyć dwa wątki. Podczas testów komputer pracował z częstotliwością 3,8 GHz.

Komputer nr 5 to najszybsza z jednostek zastosowanych do testowania badanego algorytmu. Jego procesor, Intel i7-970 ma sześć rdzeni, z których każdy obsługuje dwa wątki.

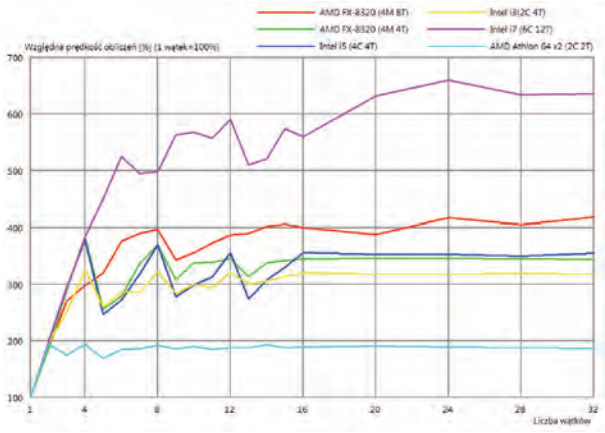
Ostatni z komputerów testowych nr 6, to maszyna z procesorem AMD Athlon 64x2, z dwoma rdzeniami i obsługującym dwa wątki.

Zmierzone zostały czasy obliczeń dla różnej liczby wątków: od 1 do 16 oraz dla 20, 24, 28 i 32 wątków. Wyniki zostały przedstawione w tab. 3 i 4 oraz na rys. 1 i 2. Minimalny czas obliczeń dla każdego z zestawów został oznaczony żółtym kolorem.

Pomiary czasów pracy badanego algorytmu były dokonywane w środowisku MS Windows. Jego procedury systemowe, wykonując się w tle, zajmują czas procesora, a co za tym idzie, wydłużają czas obliczeń. Wpływ tego zjawiska na wyniki pomiarów był istotny, różnice między dwoma uruchomieniami tego samego programu na tym samym komputerze sięgały kilku procent. W celu zminimalizowania wynikających stąd błędów pomiaru, procedura była uruchamiana 5 razy dla każdego zestawu i każdej liczby wątków, a zmierzone czasy zostały uśrednione.



Rys. 1. Czasy obliczeń w funkcji liczby wątków
Fig. 1. Computation time as a function of the number of threads



Rys. 2. Względna prędkość obliczeń w funkcji liczby wątków

Fig. 2. Relative computation speed as a function of the number of threads

Na rys. 1 i 2 można zauważyć charakterystyczne „ząbki". Wynikają one ze spadków prędkości obliczeń, których przyczyną jest nierównomierny rozdział pracy na poszczególne wątki. W przypadku wykonywania pięciu wątków na czterordzeniowym procesorze (najbardziej zauważalne spowolnienie) jeden z rdzeni musiał wykonać szeregowo dwa wątki, mając do wykonania dwukrotnie większą liczbę operacji.

5. Podsumowanie

Opracowany wielowątkowy algorytm estymacji parametrów składowych sinusoidalnych złożonego sygnału odznacza się dobrą skalowalnością. Czas przeznaczony na wykonanie procedur, które nie mogą zostać zrównoleglone, jest niewielki, a czas związany z przełączaniem danych pomijalnie mały. W tej sytuacji wydaje się być celowym opracowanie wersji tego algorytmu przystosowanej do pracy w środowiskach o większej liczbie jednostek wykonawczych, takich jak procesory GPU, a także opracowanie jego sprzętowej implementacji w układzie FPGA. Rozwiązania te, dzięki przyspieszeniu obliczeń, pozwolą być może na zastosowanie tego rodzaju algorytmu do rozwiązywania problemów praktycznych, takich jak wykrywanie anomalii i redukcja zakłóceń zawartych w sygnałach akustycznych.

Podziękowania

Publikację sfinansowano w ramach pracy własnej W/WE/8/2013.

Bibliografia

1. Zieliński T.P., *Cyfrowe przetwarzanie sygnałów,* Wydawnictwa Komunikacji i Łączności, Warszawa 2005.
2. Romberg J., Wakin M., *Compressed Sensing: A Tutorial,* IEEE Statistical Signal Processing Workshop, Madison, Wisconsin 2007.
3. Donoho D.L., *Compressed Sensing*, IEEE Transactions On Information Theory, Vol. 52, No. 4, April 2006, 1289–1306.
4. Hong S., *Direct Spectrum Sensing from Compressed Measurements*, Military Communications Conference, 2010, 1187–1192.
5. Arabas J., *Wykłady z algorytmów ewolucyjnych,* Wydawnictwa Naukowo-Techniczne, Warszawa 2001.
6. Goldberg D.E., *Algorytmy genetyczne i ich zastosowania,* Wydawnictwa Naukowo-Techniczne, Warszawa 2003.
7. Rutkowska D., Piliński M., Rutkowski L., *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte*, PWN, Warszawa, Łódź 1999.
8. Munshi A., Gaster B., Mattson T.G., *OpenCL Programming Guide*, Addison Wesley Pub Co. Inc., 2011.
9. Majewski J., Zbysiński P., *Układy FPGA w przykładach*, Wydawnictwo BTC, Warszawa 2007.
10. Kardasz P., *Rekonstrukcja zaszumionego sygnału sinusoidalnego na podstawie niewielkiej liczby próbek za pomocą algorytmu ewolucyjnego.* Pomiary Automatyka Robotyka, R. 17, nr 2/2013, 407–412.
11. Kardasz P., *Algorytm identyfikacji składowych sinusoidalnych złożonego sygnału na podstawie jego losowo pobranych próbek.* Poznan University of Technology Academic Journals. Electrical Engineering, nr 76, 2013, 197–203.

Parallel signal processing algorithm based on a small number of samples

Abstract: The paper presents a parallel algorithm for parameter estimation of sinusoidal components of a complex signal. The proposed algorithm can identify the signal components when the number of available samples of the signal is limited. The proposed algorithm was tested on test computers equipped with different number of processor cores and floating point units. The experimental results show that the proposed algorithm can work efficiently even if the number of threads exceeds the number of processor cores. Directions for further research are outlined.

Keywords: parallel algorithm, signal identification

Artykuł recenzowany, nadesłany 18.11.2013 r., przyjęty do druku 20.12.2013 r.

mgr inż. Piotr Kardasz

Doktorant Wydziału Elektrycznego Politechniki Białostockiej. Zajmuje się badaniami związanymi z inteligentnymi algorytmami kompresji, rekonstrukcji i przetwarzania sygnałów.

e-mail: pik@we.pb.edu.pl