Adam DZIEKOŃSKI, Adam LAMĘCKI, Michał MROZOWSKI

Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki

doi:10.15199/48.2016.09.03

Akceleracja metody elementów skończonych przy użyciu procesora graficznego

Streszczenie. *Artykuł przedstawia rezultaty akceleracji obliczeń metody elementów skończonych z użyciem procesora graficznego. Dzięki zastosowaniu masowo zrównoleglonych obliczeń na procesorze graficznym dwóch najbardziej kosztownych obliczeniowo etapów generacji macierzy współczynników i rozwiązywania układu równań przy użyciu metody gradientów sprzężonych z wielopoziomowym prekondycjonerem o schemacie V udało się pięciokrotnie skrócić czas symulacji metody elementów skończonych.*

Abstract. *This paper presents the results of the acceleration of computations involved in the finite element method obtained with graphics processors. A 5-fold acceleration was achieved thanks to the massive parallelization of two most time-consuming steps of the finite element method, namely matrix generation and the solution of sparse system of linear equations with the conjugate gradient method and a V-cycle multilevel preconditioner. (**Acceleration of the Finite Element Method with GPU**).*

Słowa kluczowe: procesor graficzny, metoda elementów skończonych, wielopoziomowy operator ściskający, iteracyjne metody rozwiązywania układów równań.
Keywords: GPU, FEM, multilevel preconditioner, iterative methods.

Wprowadzenie

Metoda elementów skończonych (MES) w dziedzinie częstotliwości stanowi wydajne i uniwersalne narzędzie analizy układów mikrofalowych [1, 2]. MES należy do grupy metod siatkowych, w których rozważa się różniczkową postać problemu brzegowego, zdefiniowanego w pewnym skończonym obszarze nazywanym dziedziną obliczeniową, który dzieli się na małe fragmenty poprzez wykorzystanie dedykowanych generatorów siatki. Jakość i gęstość siatki wpływa na dokładność samej symulacji. Wysoką dokładność numerycznego badanego zagadnienia elektrodynamiki obliczeniowej (ang. Computational electromagnetics, CEM) osiąga się poprzez zastosowanie siatki elementów skończonych o dużej gęstości (tzw. h-refinement) co skutkuje zwiększeniem rozmiaru macierzy opisujących dane zagadnienie CEM. Drugi sposób to podwyższenie rzędu aproksymacji wewnątrz oczka siatki (tzw. p-refinement), które także zwiększa rozmiar problemu i dodatkowo podnosi liczbę elementów niezerowych w każdym wierszu macierzy. Oba podejścia prowadzą w konsekwencji do znaczącego wzrostu kosztów numerycznych w etapie generacji i rozwiązania układów równań [3].

W literaturze znaleźć można publikacje dotyczące akceleracji obliczeń w MES. W [4] opisano przyspieszenie MES na akceleratorze graficznym (ang. Graphics Processing Unit, GPU), w którym zastosowano liniowe elementy skończone z liniowymi funkcjami bazowymi przez co koszt numeryczny jest mniejszy (mniejszy rozmiar macierzy, mniej liczby elementów niezerowych w wierszu) niż w przypadku hierarchicznych wektorowych funkcji bazowych wysokiego rzędu ($H^2(curl)$) użytych w sformułowaniach MES wykorzystanych w niniejszym artykule. W innych obszarach nauk obliczeniowych toczą się intensywne prace nad akceleracją metod numerycznych rozwiązywania problemów brzegowych i początkowych, w tym MES, a także nad rozwojem technik rozwiązywania wielkich układów równań liniowych z macierzą rzadką przy wykorzystaniu GPU. W tym ostatnim przypadku wysiłki zmierzają w kierunku znalezienia optymalnego formatu zapisu macierzy rzadkich, który z jednej strony zachowywałby zwarty zapis elementów niezerowych, a z drugiej umożliwiłaby szybki dostęp do elementów macierzy tysiącom wątków jednocześnie [5, 6, 7].

Kolejnym wątkiem naukowym ściśle związanym z formatem zapisu macierzy są iteracyjne schematy rozwiązywania układów równań liniowych. W literaturze znaleźć można publikacje opisujące implementacje technik iteracyjnych bazujących na podprzestrzeni Kryłowa [8], operatorów ściskających w postaci niekompletnej faktoryzacji LU [9] oraz metod wielosiatkowych [10], dla których uzyskano znaczące skrócenia czasu rozwiązania dzięki zastosowaniu GPU. Dostępne są również biblioteki dedykowane zagadnieniu rozwiązywania układów równań na GPU, np. MAGMA Sparse-Iter Package [11], AmgX [12]. Należy podkreślić, że wiele technik iteracyjnego rozwiązywania układów równań liniowych nie nadaje się dla systemów powstających w CEM ze względu na złe uwarunkowanie generowanych macierzy, co przekłada się na słabą zbieżność lub jej brak.

Wątek szybkiego wyznaczania macierzy współczynników w MES jest istotny dla wielu obszarów nauk obliczeniowych. MES jest jedną z najbardziej popularnych technik symulacyjnych wykorzystywanych w mechanice płynów i mechanice stosowanej i dlatego w literaturze znaleźć można publikacje dotyczące najważniejszych etapów budowania macierzy rzadkich w MES (całkowanie numeryczne w celu wyznaczenia lokalnych macierzy elementów [13], składanie macierzy rzadkiej z wykorzystaniem GPU [4, 14-18]).

W tym artykule przedstawiono wyniki akceleracji metody elementów skończonych uzyskanej dzięki zastosowaniu procesora graficznego (GPU) do wykonania obliczeń numerycznych. W sekcji 2 przedstawiono sformułowania metody elementów skończonych użyte w artykule. W następnej sekcji przedstawiono charakterystykę i model programowania na GPU. W kolejnej sekcji przedstawiono listę optymalizacji etapów generacji i rozwiązywania układu równań na GPU. W sekcji 5 umieszczono wyniki testów numerycznych i otrzymane wnioski. Artykuł kończy podsumowanie.

Sformułowania metody elementów skończonych

W niniejszym artykule MES wykorzystana jest do wyznaczenia parametrów obwodowych układu mikrofalowego. W odniesieniu do opisu obwodowego wykorzystano sformułowania zaproponowane w [19], które bazują na technice segmentacji obwodu. Dobrym sposobem implementacji metody segmentacji jest opis każdego regionu poprzez wielorodzajową macierz układu wielowrotowego, która w połączeniu z innymi macierzami z pozostałych regionów pozwala otrzymać globalną

odpowiedz układu. Jednym z możliwych sposobów obliczenia tej macierzy jest koncepcja uogólnionej macierzy impedancji (ang. generalized impedance matrix, GIM) [19]. Na podstawie otrzymanej macierzy impedancji następnie wyznacza się macierze parametrów rozproszenia $\mathbf{S}_C$. Elementy macierzy rozproszenia są funkcją częstotliwości. Znając wartości elementów macierzy $\mathbf{S}_C$ w analizowanym paśmie częstotliwości określa się charakterystyki odbicia i transmisji analizowanego układu.

W artykule rozwiązywane jest równanie fali Helmholtza:

$$\nabla \times \mu^{-1} \nabla \times \vec{E} - \omega^2 \varepsilon \vec{E} = 0 \tag{1}$$

gdzie $\vec{E}$ to wektor natężenia pole elektrycznego, ω reprezentuje częstotliwość kątową zależną od częstotliwości f, ε i μ opisują tensory przenikalność elektrycznej i magnetycznej.

Słabą formę wektorowego równania Helmholtza można wyprowadzić przy użyciu metody Galerkina poprzez przemnożenie równania (1) przez wektorowe funkcje wagi $\vec{W}$. W metodzie Galerkina funkcje wagi $\vec{W}$ wybiera się spośród zestawu funkcji bazowych (w sformułowaniu użytym w artykule do opisu elementu skończonego użyto hierarchiczne funkcje bazowe).

Uogólnioną macierz impedancji (GIM) można zapisać w postaci równania (2) (dokładne wyprowadzenie w [19]). Macierz $\mathbf{B}$ reprezentuje pobudzenie, a macierz $\mathbf{X}$ to rozwiązanie układu równań liniowych (3), które interpretuje się jako macierz poszukiwanych amplitud funkcji bazowych opisujących falę elektromagnetyczną.

$$Z(j\omega) = j\omega\, \mu_0 \mathbf{B}_n^T \mathbf{A}^{-1} \mathbf{B}_n = j\omega\, \mu_0 \mathbf{B}_n^T \mathbf{X} \tag{2}$$

$$\mathbf{AX} = \mathbf{B}_n \tag{3}$$

Macierz rozproszenia można zapisać w postaci równania (4), gdzie macierz admitancji $\mathbf{Y}$ jest odwrotnością macierzy impedancji ($\mathbf{Y} = \mathbf{Z}^{-1}$). Na podstawie wartości macierzy rozproszenia można określić współczynniki odbicia sc_{11} i transmisji sc_{21} układów mikrofalowych w analizowanym paśmie częstotliwości:

$$[\mathbf{S}_C] = 2(\mathbf{I}_d + \mathbf{Y})^{-1} - \mathbf{I}_d \tag{4}$$

W przypadku macierzy wygenerowanej przy użyciu sformułowań MES w analizowanych zagadnieniach macierz rzadka $(\mathbf{A} = \mathbf{S} - k_0^2\mathbf{T})$ jest symetryczna, ale jest nieokreślona (ang. indefinite). Jest to spowodowane faktem, że jej składniki mają odmienny charakter, tzn. macierz sztywności $\mathbf{S}$ jest półdodatnio określona (ang. positive semidefinite), a przeskalowana macierz bezwładności $-k_0^2\mathbf{T}$ jest ujemnie określona ponieważ ma ujemne wartości własne (k_0 - liczba falowa). Dodatnia określoność macierzy gwarantuje, że podczas obliczeń w procesie iteracyjnym nie wystąpią nieprawidłowe operacje (dzielenie przez zero), jednakże w praktyce taka sytuacja nie występuje mimo, że macierz jest nieokreślona i dlatego metoda gradientów sprzężonych (przyp. dedykowana dla macierzy dodatnio określonych) jest wykorzystywana w analizie MES problemów elektromagnetycznych. W celu zagwarantowania zbieżności procesu iteracyjnego zastosowano wielopoziomowy operator ściskający o cyklu V (oznaczone w dalszej części artykułu jako PCG-V) [3].

Programowanie akceleratorów graficznych

W niniejszej sekcji przedstawiono charakterystykę i model programowania w architekturze CUDA. W GPU kompatybilnych z architekturą CUDA występuje zupełnie inna organizacja obliczeń i pamięci niż w procesorze centralnym (ang. Central Processing Unit, CPU) [3, 20, 21]. CPU korzysta z pamięci zewnętrznej (DRAM), podręcznej (cache) i rejestrów. Na GPU procesory obliczeniowe grupowane są w tzw. multiprocesory. W architekturze Kepler jeden multiprocesor zawiera 192 rdzeni. Multiprocesor może korzystać z pamięci: globalnej (ang. global memory), wspólnej (ang. shared memory), tekstur (ang. texture memory), stałej (ang. constant memory) i rejestrów. W akceleratorach graficznych kompatybilnych z architekturą CUDA występuje inny rozkład tranzystorów do poszczególnych bloków funkcjonalnych niż na CPU. Na GPU więcej tranzystorów przeznaczonych jest na jednostki arytmetyczno-logiczne niż na sterowanie i pamięci podręczne, co pozwala na uzyskanie dużej mocy obliczeniowej [3, 20, 21].

W dalszej części tej sekcji przedstawiono podstawowy model programowania w architekturze CUDA. Najpierw dane wejściowe algorytmu alokowane są na CPU, skąd kopiowane są na GPU. Zapisując dane w pamięci GPU do wyboru są pamięć globalna, tekstur i stała. Następnie z poziomu CPU wywoływany jest program wykonywany na GPU (tzw. kernel), w trakcie którego wątki mogą odczytywać dane z powyżej wymienionych trzech rodzajów pamięci. Obliczenia w kernelu wykonywane są przez wątki (ang. threads), które pracują równolegle, tzn. wykonują równolegle te same instrukcje na różnych danych (ang. Single Instruction Multiple Threads, SIMT). Wątki grupowane są w bloki wątków (ang. blocks), a bloki uporządkowane są w siatki bloków (ang. grids). Po zakończeniu obliczeń w kernelu należy dane wynikowe zapisać do pamięci globalnej, skąd po wykonaniu kernela mogą zostać skopiowane z GPU na CPU.

Na zakończenie tej sekcji trzeba zaznaczyć, że architektura CUDA umożliwia implementację algorytmów z masowo zrównoleglonymi obliczeniami, jednakże nie każdy rodzaj algorytmu posiada cechy, które pozwalają na jego efektywną implementację na GPU. W niektórych rodzajach algorytmów kod wykonywany na GPU jest wolniejszy niż jego odpowiednik na CPU. Mając to na uwadze, programując karty graficzne należy podjąć trud maksymalnego zrównoleglenia algorytmów, optymalizować użycie pamięci, zwracać uwagę by używać optymalnych instrukcji. Często wieloetapowa optymalizacja wybranego algorytmu prowadzi do opracowania zupełnie nowego algorytmu z masowo zrównoleglonymi obliczeniami, który realizuje tę samą funkcjonalność co algorytm początkowy.

Akceleracja metody elementów skończonych

Implementacja generacji macierzy rzadkich i rozwiązywania układu równań zostały dogłębnie opisane w [22-24]. Poniżej przedstawione zostaną najważniejsze cechy akceleracji obliczeń na GPU. Generacja została podzielona na trzy etapy. W pierwszym z nich wykonywane jest całkowanie numeryczne celem wyznaczenia lokalnych macierzy elementów przy użyciu wielopunktowej kwadratury Gaussa. Następnie występuje etap składania macierzy w formacie COO (ang. Coordinate format), by potem przekonwertować macierz do formatu CRS (ang. Copressed Row Storage) wraz z eliminacją duplikatów [22].

W etapie całkowania numerycznego zaproponowano wielopoziomowe zrównoleglenie obliczeń, m.in. w każdej iteracji generacji macierzy podzbiór czworościanów jest przetwarzany w taki sposób, iż porcja czworościanów (256) jest przetwarzana równolegle (na poziomie bloków wątków), dla każdego czworościanu obliczenia związane z kwadraturą Gaussa są zrównoleglone (na poziomie bloków wątków), operacje na macierzach gęstych (mnożenie

MOST WIEDZY Pobrano z mostwiedzy.pl

macierzy, dodawanie macierzy) w każdym punkcie kwadratury są zrównoleglone (na poziomie wątków), współbieżne strumienie (ang. concurrent streams, Hyper-Q) są w zależności od ośrodka przyporządkowane do oddzielnych wariantów całkowania numerycznego. Obliczenia w etapie konstrukcji macierzy również zostały zrównoleglone, w szczególności podczas składanie macierzy w formacie COO jeden wątek odpowiada za wpisanie do macierzy rzadkiej jednej trójki opisującej element niezerowy macierzy (indeks wiersz, indeks kolumny, wartość niezerowa). Na potrzebę konwersji macierzy z formatu COO do formatu CRS z eliminacją duplikatów opracowano dedykowany algorytm.

W kontekście rozwiązywania układów równań w MES opracowano hybrydową implementację iteracyjnej metody gradientów sprzężonych z wielopoziomowym operatorem ściskającym o schemacie V pozwalającą rozwiązać układ równań liniowych z wykorzystaniem pojedynczego akceleratora. W szczególności opracowano nowy format zapisu macierzy rzadkiej (Sliced ELLR-T) dedykowany wykonaniu operacji mnożenia macierzy rzadkiej przez wektor na GPU, który jest wydajniejszy niż implementacje dostępne w bibliotece CUSPARSE oparte na formacie CRS [23]. Dodatkowo, zaproponowano zastąpienie metody Gaussa-Seidela przez ważoną metodę Jacobiego w operacjach „wygładzających" wielopoziomowego operatora ściskającego [24].

Rezultaty numeryczne

Testy numeryczne etapów generacji i rozwiązywania układów równań (omówione w kolejnych sekcjach) przeprowadzono dla filtru grzebieniowego 9 rzędu pracującego w paśmie GSM (920-980 MHz) [3, 22]. Czworościany w strukturze wypełnione są izotropowymi ośrodkami bezstratnymi, w których względna przenikalność elektryczna ε = 1 i względna przenikalność magnetyczna μ = 1. Jako warunki brzegowe zastosowano PEC (ang. Perfect Electric Conductor). Filtr został pobudzony z linii współosiowej falą o rodzaju TEM (ang. Transverse Electric-Magnetic). Testy numeryczne wykonano dla serwera obliczeniowego: GPU (Tesla K40, 2880 rdzeni, 12GB), CPU (Intel E5-2687W, 8 rdzeni, 256 GB RAM).

Opracowane strategie zrównoleglenia obliczeń pozwoliły uzyskać znaczne skrócenie czasu generacji macierzy względem CPU: 7,5 krotne przyspieszenie całkowania numerycznego, 15,5 krotne przyspieszenie budowy macierzy w formacie COO i 30 krotne skrócenie czasu konwersji macierzy do formatu CRS, co przełożyło się na ok. 11 krotną redukcję czasu generacji macierzy sztywności i bezwładności, względem zoptymalizowanej i zrównoleglonej implementacji na CPU.

Zastosowanie GPU pozwoliło na ok. 3 krotne przyspieszenie iteracyjnego rozwiązywania układu równań względem implementacji metody iteracyjnej na CPU. Dodatkowo, przyspieszenie nieznacznie rośnie wraz ze wzrostem rozmiaru problemu. Ponadto, otrzymane wyniki wskazują, że metoda bezpośrednia (Intel MKL Pardiso) jest wydajniejsza dla mniejszych macierzy – jest to spowodowane tym, że dla mniejszych problemów faktoryzacja numeryczna jest krótsza niż czas iteracyjnego rozwiązywania układu równań. Dla macierzy o rozmiarze powyżej 2 milionów wierszy zastosowanie akceleratora do wykonania obliczeń w metodzie gradientów sprzężonych z wielopoziomowym operatorem ściskającym pozwala uzyskać przyspieszenie względem metody bezpośredniej. Jest to spowodowane tym, że użycie GPU sprawia, że rozwiązanie układu trwa krócej niż faktoryzacja numeryczna.

Wysiłek włożony w zrównoleglenie generacji macierzy oraz implementacji iteracyjnej metody rozwiązywania układów równań pozwala na znaczące ok. 4,7 krotne skrócenie czasu całej analizy metody elementów skończonych. Gdy liczba częstotliwości w paśmie wynosi 100 czas analizy MES został skrócony z 8,5 godzin do 100 minut.

Podsumowanie

Zaproponowane implementacje pozwoliły na skrócenie czasu obliczeń najbardziej kosztownych obliczeniowo etapów metody elementów skończonych, czyli generacji globalnych macierzy współczynników oraz rozwiązania układu równań liniowych. Zastosowanie GPU do wykonania obliczeń najbardziej kosztownych obliczeniowo etapów MES pozwoliło na ok. 4,7 krotne skrócenie czasu analizy MES (problem 5 milionów niewiadomych). Czas analizy MES został skrócony z niespełna pięciu godzin (gdy obliczenia wykonywane są wyłącznie na CPU) do ok. godziny gdy obliczenia w etapach generacji macierzy i iteracyjnego rozwiązania układu równań wykonywane są na GPU.

Praca wykonana w ramach działalności statutowej oraz WiComm's GPU Research Center for Computational Electromagnetics na Politechnice Gdańskiej.

Autorzy: *dr inż. Adam Dziekoński, dr inż. Adam Lamęcki, prof. dr hab. inż. Michał Mrozowski, Politechnika Gdańska, Wydział Elektroniki, Telekomunikacji i Informatyki, ul. Gabriela Narutowicza 11/12, 80-233 Gdańsk, E-mail: adam.dziekonski@gmail.com, adlam@eti.pg.gda.pl, michal.mrozowski@eti.pg.gda.pl.*

LITERATURA

[1] Jin J., The Finite Element Method in Electromagnetics, John Wiley and Sons Inc., 2002
[2] Volakis J.L., Chatterjee A., Kempel L.C., Finite Element Method for Electromagnetics. Antennas, Microwave Circuits and Scattering Applications, *IEEE Series on Electromagnetic Wave Theory*, 1998
[3] Dziekonski A., Optymalizacja wydajności obliczeniowej metody elementów skończonych w architekturze CUDA, rozprawa doktorska, Politechnika Gdańska, 2015
[4] Meng H.-T., Nie B.-L., Wong S., Macon C., Jin J.-M., GPU Accelerated Finite Element Computation for Electromagnetic Analysis, *IEEE Antennas Propag. Mag,* vol. 56, (2014) no. 2, 39-62
[5] Kreutzer M., Hager G., Wellein G., Fehske H., Bishop A.R., A Unified Sparse Matrix Data Format for Efficient General Sparse Matrix-Vector Multiplication on Modern Processors with Wide SIMD Units, *SIAM Journal on Scientific Computing,* vol. 36, (2014) no. 5, C401-C423
[6] Anzt H., Tomov S., Dongarra J., Implementing a Sparse Matrix Vector Product for the SELL-C\SELL-C-σ formats on NVIDIA GPUs. Raport instytutowy, University of Tennessee, Department of Electrical Engeneering & Computer Science, 2014
[7] Langr D., Tvrdik P., Evaluation Criteria for Sparse Matrix Storage Formats, *IEEE Transactions on Parallel and Distributed Systems*, vol.27, no.2, 428-440, Feb. 1 2016
[8] Li R., Saad Y., GPU-Accelerated Preconditioned Iterative Linear Solvers, *Journal of Supercomputing*, vol. 63, (2013) no. 2, 443-466
[9] Naumov M., Incomplete-LU and Cholesky Preconditioned Iterative Methods Using CUSPARSE and CUBLAS, http://docs.nvidia.com/cuda/incomplete-lu-cholesky/
[10] Goeddeke D., Strzodka R., Mohd-Yusof J., McCormick P., Wobker H., Becker C., Turek S., Using GPUs to Improve Multigrid Solver Performance on a Cluster, *International Journal on Computer Science and Engineering,* vol. 4, (2008) no. 1, 36-55
[11] Matrix Algebra on GPU and Multicore Architectures, http://icl.cs.utk.edu/ magma/index.html
[12] AmgX, https://developer.nvidia.com/amgx
[13] Banas K., Plaszewski P., Maciol P., Numerical Integration on GPUs for Higher Order Finite Elements. *Computers and*

MOST WIEDZY Pobrano z mostwiedzy.pl

Mathematics with Applications, vol. 67, (2014) 1319-1344
[14] Markall G., Slemmer A., Ham D., Kelly P., Cantwell C., Sherwin S., Finite Element Assembly Strategies on Multi-Core and Many-Core Architectures, *Int. J. Numer. Meth. Fluids,* vol. 71, (2013) no. 1, 80-97
[15] Cecka C., Lew A.J., Darve E., Assembly of Finite Element Methods on Graphics Processors, *International Journal for Numerical Methods in Engineering*, vol. 85, (2011) no. 3, 640-669
[16] Georgescu S., Chow P., Okuda H., GPU Acceleration for FEM-Based Structural Analysis, *Archives of Computational Methods in Engineering,* vol. 20, (2013) no. 2, 111-121
[17] Reguly I.Z., Giles M.B., Finite Element Algorithms and Data Structures on Graphical Processing Units, *International Journal of Parallel Programming,* vol. 43, (2015) no. 2, 203-239
[18] Fu Z., Lewis T.J., Kirby R.M., Whitaker R.T., Architecting the Finite Element Method Pipeline for the GPU, *Journal of Computational and Applied Mathematics,* vol. 257, (2014) 195-211
[19] Rubio J., Arroyo J., Zapata J., Analysis of Passive Microwave Circuits by Using a Hybrid 2-D and 3-D Finite-Element Mode-Matching Method, *IEEE Trans. Microw. Theory Techn.,* vol. 47, (1999) no. 9, 1746-1749
[20] Sanders J., Kandrot E., CUDA by Example: An Introduction to General-Purpose GPU Programming, NVIDIA Corporation, 2011
[21] Nvidia. CUDA Programming Guide Version 4.0. Nvidia, czerwiec 2011. http://www.nvidia.com/object/cuda_develop.html.
[22] Dziekonski A., Sypek P., Lamecki A., Mrozowski M., Generation of Large Finite Element Matrices on Multiple Graphics Processors, *Int. J. Numer. Meth. Eng.,* vol. 94, (2012)no. 2, 204-220
[23] A. Dziekonski, A. Lamecki, M. Mrozowski, A Memory Efficient and Fast Sparse Matrix Vector Product on a GPU, *Progress In Electromagnetics Research*, vol. 116, (2011) 49-63
[24] A. Dziekonski, A. Lamecki, M. Mrozowski, GPU Acceleration of Multilevel Solversfor Analysis of Microwave Components With Finite Element Method, *IEEE Microw. Compon. Lett*., vol. 1, (2011) 1-3

MOST WIEDZY Pobrano z mostwiedzy.pl