Rafał RÓŻYCKI, Grzegorz WALIGÓRA, Jan WĘGLARZ
Politechnika Poznańska

MODELE WYKONYWANIA ZADAŃ OBLICZENIOWYCH W ENERGOOSZCZĘDNYCH SYSTEMACH KOMPUTEROWYCH

Streszczenie. W niniejszej pracy rozważane są nowoczesne architektury systemów komputerowych oraz adekwatne dla nich modele wykonywania zadań obliczeniowych. Opisano i porównano dwa podstawowe modele oraz wskazano w jakich sytuacjach są one użyteczne. Oba modele opracowane zostały na potrzeby technologii CMOS powszechnie dzisiaj stosowanej do budowania układów mikroprocesorowych.

MODELS OF PROCESSING COMPUTATIONAL TASKS IN ENERGY–AWARE COMPUTING SYSTEMS

Summary. In this work modern architectures of computing systems as well as related models of processing computational tasks are considered. Two basic processing models are compared, and their applicability and usefulness in practical situations are discussed. The models have been elaborated for the CMOS technology, commonly used nowadays to build microprocessor systems.

1. Wstęp

Powszechnie stosowaną techniką zmniejszania poboru mocy w nowoczesnych procesorach CMOS [4] jest zmniejszanie częstotliwości taktowania zegara w okresie, gdy procesor nie jest używany do intensywnych obliczeń. Technika ta nazywana jest dynamicznym skalowaniem częstotliwości (ang. *Dynamic Frequency Scaling* – DFS). Obok redukcji poboru mocy przyczynia się ona również do zmniejszenia ryzyka przegrzania procesora, jednakże nie powoduje obniżenia zużycia energii. Zmniejszenie zużycia energii możliwe jest dzięki technice zwanej dynamicznym skalowaniem napięcia (ang. *Dynamic Voltage Scaling* – DVS). Technika DVS poprawia wprawdzie zużycie energii przez procesor, jednak czyni to kosztem jego szczytowych możliwości obliczeniowych. Jeśli jednak w odpowiedzi na zmieniające się obciążenie procesora użyć techniki DVS w kombinacji z DFS, to zarówno zużycie energii, jak i efektywność przetwarzania mogą być poprawione. Właśnie takie podejście najczęściej realizowane jest przez procesory o zmiennej szybkości przetwarzania (ang. *Variable Speed Processor* – VSP). Zostały one zaprojektowane w celu zmniejszenia zużycia energii w trakcie przetwarzania przez procesor niekrytycznych zadań obliczeniowych i w celu ochrony systemu komputerowego przed przegrzaniem. Przez

zadanie obliczeniowe rozumieć będziemy proces systemu komputerowego (najczęściej aplikację komputerową), który stanowi odrębną jednostkę, przy czym za przydział zasobów niezbędnych do jego wykonania odpowiada system operacyjny. Jako że zadanie obliczeniowe może być przetwarzane w systemach wyposażonych w VSP z różną szybkością, niezbędne jest uwzględnienie tego faktu w tzw. modelu wykonywania zadania.

Na dobór odpowiedniego modelu wykonywania zadania obliczeniowego niebagatelny wpływ ma architektura procesora, na którym ma być ono wykonywane. Poniżej przedstawimy koncepcje architektur procesorów, które realizują postulat oszczędnego gospodarowania mocą i energią w bardziej wyrafinowany sposób niż standardowe procesory VSP.

2. Nowoczesne architektury systemów komputerowych

Współczesne procesory zbudowane z milionów tranzystorów zawierają również skomplikowane układy kombinacyjne. Dla każdej instrukcji procesora czas przepływu sygnału przez układ kombinacyjny może być inny. Maksymalna szybkość przetwarzania procesora, dla danego napięcia zasilania, jest zwykle zdeterminowana przez długość najdłuższej ścieżki (ścieżki krytycznej) przepływu sygnału w takich układach. Architekturę, która wykorzystuje możliwość dostosowywania częstotliwości i napięcia zasilania w każdym cyklu pracy procesora zaproponowano w [1]. Dzięki niewielkim zmianom wprowadzonym w projekcie standardowego procesora synchronicznego, możliwe stało się zwiększenie szybkości przetwarzania przy niezmienionym zużyciu energii lub zachowanie szybkości obliczeń z jednoczesnym oszczędzaniem energii. Zastosowane rozwiązania umożliwiają wykonanie każdej instrukcji w czasie do niej dobranym, wystarczającym do ustalenia się sygnałów wyjściowych w układach kombinacyjnych procesora, a nie w czasie dostosowanym do najwolniejszej instrukcji. W procesorach potokowych do predykcji najdłuższego czasu propagacji sygnałów dla kolejnych operacji wykonywanych przez procesor wykorzystane mogą być kody instrukcji oraz sygnały z potoku. Opóźnienia dla każdej instrukcji, przy różnych napięciach zasilania zmierzyć można doświadczalnie na etapie konstruowania procesora. Wartości te mogą być następnie zapisane w lokalnej "pamięci opóźnień" i używane w czasie normalnej pracy procesora do ustalania aktualnej długości okresu jego zegara. Chociaż autorzy projektu uwzględnili w nim możliwość zastosowania mechanizmu dynamicznego skalowania napięcia, eksperymenty przeprowadzono sterując stałym, wspólnym dla wszystkich instrukcji napięciem zasilania. W rezultacie zbudowany system wykazał większą szybkość obliczeń przy tym samym zużyciu energii i oszczędność energii przy tej samej szybkości w porównaniu do procesora wykonanego w standardowej architekturze synchronicznej.

Chociaż przedstawiona wyżej koncepcja architektury procesora jest obiecująca z punktu widzenia oszczędnego gospodarowania energią w systemach komputerowych, ma też jednak swoje wady. Długość okresu zegara systemowego dopasowywana jest do pojedynczej instrukcji, bez uwzględniania informacji na wyższym poziomie abstrakcji – na poziomie aplikacji, w ramach której jest realizowana. Oczywiście trudno na poziomie sprzętowym ocenić, czy dana instrukcja jest częścią aplikacji

obliczeniowo wymagającej, czy też nie. Z drugiej strony, w systemie wielozadaniowym sterowanie szybkością przetwarzania procesora nie może odbywać się z poziomu samej aplikacji, gdyż jest ona nieświadoma wymagań dotyczących przetwarzania innych aplikacji. W rezultacie naturalne jest, że sterowanie szybkością przetwarzania w celu ograniczania zużycia energii powinno odbywać się na poziomie systemu operacyjnego.

Propozycję architektury VSP dla systemów mikroprocesorowych ogólnego przeznaczenia zaproponowano w [2]. W projekcie tym założono współdziałanie trzech głównych komponentów:

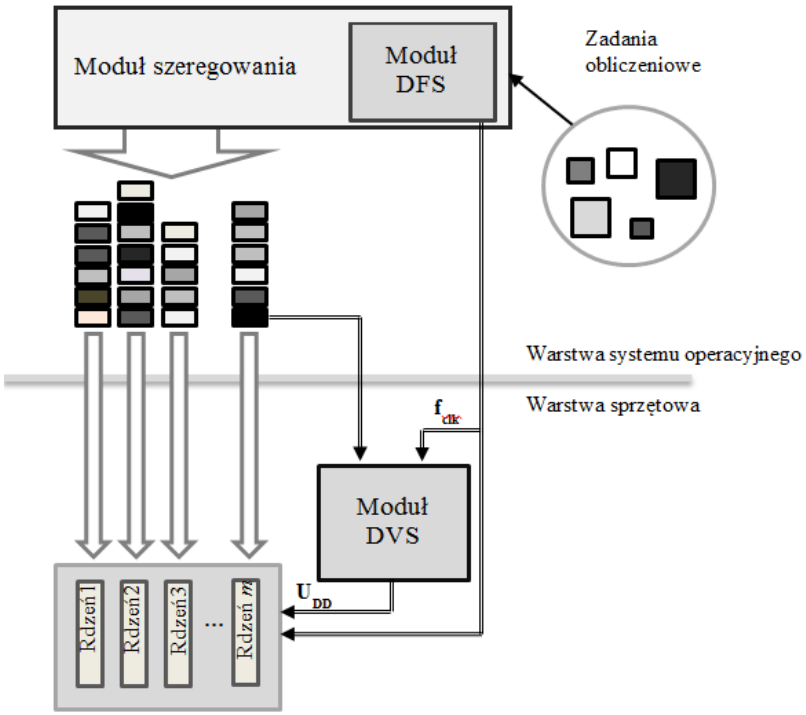
- systemu operacyjnego, który w sposób inteligentny może zmieniać szybkość procesora
- pętli regulacji (ang. *regulation loop*), która generuje minimalne napięcie dla danej szybkości
- procesora, który jest w stanie działać w szerokim zakresie wartości napięcia zasilania.

Autorzy projektu założyli, że aplikacje komputerowe są w stanie udostępniać pewne użyteczne informacje na temat ich wymagań dotyczących przetwarzania w danym systemie komputerowym. Taka informacja może zawierać np.: pożądany czas zakończenia, pożądaną szybkość przetwarzania (podawaną jako częstotliwość zegara w MHz), liczbę klatek na sekundę odtwarzanego obrazu, itd., itp. Bazując na takiej informacji oraz stanie obliczeń na poziomie całego systemu komputerowego, system operacyjny wybiera szybkość przetwarzania każdej aplikacji. Aby uzyskać efekt oszczędzania energii, każdej zmianie szybkości procesora odpowiada zmiana napięcia zasilania. Zakłada się jednak, że system operacyjny jest nieświadomy, jak w sposób optymalny należy dobrać napięcie zasilania do poszczególnych aplikacji. To zadanie realizowane jest na poziomie sprzętowym. Algorytm zaimplementowany w ramach modułu sterującego szybkością przetwarzania dobiera optymalną częstotliwość zegara na bazie informacji dostarczanych przez wszystkie aplikacje w systemie, uwzględniając przy tym ograniczenia dotyczące maksymalnych opóźnień uszeregowanych zadań. W rezultacie system operacyjny zawsze ustawia szybkość przetwarzania na najniższym dopuszczalnym poziomie akceptowalnym przez aktualnie uruchomione aplikacje, zapewniając jednocześnie minimalizację zużycia energii. Zaproponowany prototypowy system procesorowy pokazał, że taka koncepcja DVS może wyraźnie poprawić czas pracy systemów zasilanych akumulatorowo, bez pogorszenia efektywności przetwarzania samego systemu. Moc obliczeniowa i zużyta energia wykorzystywane do realizacji zadania sterowania szybkością przetwarzania na poziomie systemu operacyjnego są przy tym pomijalnie małe.

3. Idea wielordzeniowego procesora DVS

Architektury opisane powyżej mogą być bazą do opracowania idei architektury wielordzeniowego procesora o zmiennej szybkości przetwarzania [5] (por. rys. 1). W architekturze tej zakłada się, że każdy z rdzeni procesora wielordzeniowego może mieć indywidualnie dobieraną częstotliwość pracy f_{clk} oraz napięcie zasilania U_{DD}. Moduł szeregujący spełnia dwie funkcje: określa porządek wykonania poszczególnych

zadań obliczeniowych (lub ich fragmentów) na poszczególnych rdzeniach i dobiera do uporządkowanych zadań wektor właściwych częstotliwości zegara na rdzeniach (za pomocą modułu DFS). Sprzętowy moduł DVS na podstawie określonej wcześniej częstotliwości zegara oraz rodzaju wykonywanej instrukcji dobiera właściwe napięcie zasilania dla każdego rdzenia. Dzięki takiej architekturze otrzymujemy najbardziej elastyczny model przetwarzania zadań obliczeniowych, w którym szybkość przetwarzania oraz pobierana moc przez rdzeń procesora zależą od wykonywanego zadania obliczeniowego, a nie tylko od konkretnej instrukcji wykonywanej w ramach takiego zadania. W modelu tym moduł szeregujący systemu operacyjnego jest w pełni odpowiedzialny za aspekt energetyczny wykonywania zbioru zadań obliczeniowych i dlatego winien być wyposażony w mechanizmy pozwalające na optymalizację wybranych kryteriów szeregowania. Mechanizmy takie opracowuje się np. dzięki formułowaniu odpowiedniego problemu optymalizacyjnego w postaci wielomaszynowego problemu szeregowania zadań z dodatkowymi ograniczeniami zasobowymi.



Rys.1. Idea architektury wielordzeniowego procesora VSP

4. Modele wykonywania zadania

4.1. Model klasyczny

W rozdziałach 2 i 3 przedstawione zostały architektury systemów komputerowych, które umożliwiają wykonywanie zadań obliczeniowych z różną szybkością. Podstawowym zagadnieniem staje się więc sposób reprezentowania zależności między czasem (szybkością) wykonywania zadania obliczeniowego, a pobieraną do tego celu mocą lub zużywaną energią. Za pomocą modelu wykonywania zadania wyraża się ten związek w sposób formalny.

Powszechnie stosowanym modelem wykonywania zadania uwzględniającym aspekt energetyczny jest model wykorzystujący **funkcję poboru mocy** (ang. *power usage function*) postaci:

$$p(s) = s^{\alpha}, \ \alpha > 1 \tag{1}$$

gdzie s oznacza szybkość procesora wykonującego zadanie, a $p(s)$ oznacza wynikający z tej szybkości pobór mocy. Dla procesorów synchronicznych wykonanych w technologii CMOS powszechnie przyjmuje się, że α jest równe 3.

Z równania (1) wynika, że funkcja poboru mocy jest ściśle wypukła. Oznacza to, że w celu minimalizacji całkowitej energii zużywanej przez procesor do wykonania danego zadania obliczeniowego należy wykonywać je z możliwie najmniejszą szybkością. W modelu tym przyjmuje się też, że zadanie i charakteryzuje parametr w_i ($w_i > 0$) oznaczający liczbę cykli procesora niezbędnych do wykonania całego zadania i. Parametr w_i nazywać będziemy **rozmiarem zadania** i. Oczywiście czas wykonania zadania i zależy od jego rozmiaru w_i i szybkości procesora wykonującego to zadanie. Zadanie i rozpoczęte w chwili a_i uważa się za zakończone w chwili C_i jeśli spełniony jest następujący warunek:

$$\int_{a_i}^{C_i} s(t)dt = w_i \tag{2}$$

Wykonanie zadania i w przedziale czasu $[a_i, C_i]$ wymaga zużycia pewnej ilości energii E_i, którą to ilość można wyznaczyć jako:

$$E_i = \int_{a_i}^{C_i} p(s(t))dt \tag{3}$$

Należy zwrócić uwagę, że w modelu (1) szybkość przetwarzania procesora traktowana jest jako zmienna decyzyjna, która wpływa na chwilowe zużycie energii, czyli moc. Na argument funkcji w modelu (1) nie nakłada się zwykle górnego ograniczenia, tj. przyjmuje się, że $s \in [0, \infty)$. W konsekwencji powiązana z szybkością przetwarzania moc jest również nieograniczona z góry. Oczywiście ilość pobieranej przez procesor mocy bezpośrednio wpływa na ilość ciepła wydzielaną przez niego podczas przetwarzania zadań obliczeniowych. Nadmierny wzrost temperatury układu mikroprocesorowego doprowadzić może do jego wadliwego funkcjonowania, a nawet trwałego uszkodzenia.

Na bazie modelu (1) zaproponowano w literaturze dwie główne klasy deterministycznych problemów szeregowania zadań uwzględniających aspekty energetyczne. Pierwsza klasa obejmuje przypadki, w których minimalizowana jest ustalona czasowa miara jakości uszeregowania przy założeniu ograniczonej ilości dostępnej energii. W drugim przypadku poszukuje się rozwiązań, które minimalizują zużywaną energię przy jednoczesnym spełnieniu przez poszukiwane uszeregowanie określonych wymagań dotyczących jego jakości. Pierwsza klasa problemów, określana mianem klasy **problemów laptopowych** (ang. *laptop problem*), zawiera przypadki typowe dla komputerów przenośnych zasilanych energią zgromadzoną w akumulatorach o określonej pojemności. Z drugą klasą problemów, klasą **problemów serwerowych** (ang. *server problem*), mamy na przykład do czynienia, gdy pożądane jest zmniejszenie kosztów zużycia energii przez system komputerowy na stałe podłączony do źródła zasilania przy jednoczesnym utrzymaniu określonego poziomu efektywności obliczeń.

4.2. Model dynamiczny

Uzależnianie czasu wykonania zadania od maszyny, na której jest ono wykonywane (jak ma to np. miejsce w przypadku problemów szeregowania na maszynach dowolnych) jest niewystarczające w przypadku, gdy na czas wykonania zadania mają wpływ również dodatkowe zasoby. W takiej sytuacji stosuje się modele zadań, które w sposób najbardziej adekwatny odzwierciedlają naturę przyjętych zasobów. Jeśli przyjąć, że dodatkowe zasoby mogą być przydzielane jedynie w liczbach jednostek określonych przez skończone zbiory dopuszczalnych wartości, to najwygodniej jest stosować tzw. **tryby wykonywania zadań** (ang. *processing modes*). Każdy tryb zawiera informację o liczbie jednostek zasobów wykorzystywanych przez zadanie (w tym numer maszyny, gdy maszyny nie są identyczne) i powiązany z tym trybem czas wykonywania zadania. Wadą takiego podejścia jest to, że każde zadanie wykonywane może być jedynie w tym trybie, w którym zostało rozpoczęte. Oznacza to, że zmiana przydziału zasobów w trakcie wykonywania zadania jest niedopuszczalna. Ponadto, założenie dyskretnych przydziałów zasobów w modelu z trybami wykonywania nie pozwala na wykorzystanie ciągłej natury niektórych zasobów dodatkowych, takich jak np. energia czy moc, które przydzielane mogą być w dowolnych ilościach z pewnego przedziału, a nie zbioru wartości dopuszczalnych.

Powyższych wad nie posiada **dynamiczny model** wykonywania zadania zaproponowany przez Burkova w [3], a następnie rozwijany w wielu pracach przez Węglarza. W pracy [6] rozważano problem rozdziału zasobu ciągłego, odnawialnego, w którym chwilowa szybkość wykonywania zadania zależy od aktualnego przydziału tego zasobu. Jeśli założyć, że zasobem ciągłym jest limitowana moc, wtedy chwilową szybkość wykonywania zadania obliczeniowego w sposób formalny wyrazić można za pomocą **funkcji szybkości wykonywania** (w skrócie – **funkcji szybkości**) w następujący sposób:

$$\dot{x}(t) = \frac{dx_i(t)}{dt} = s_i(p_i(t)), \;\; x_i(0) = 0; \; x_i(C_i) = w_i \tag{4}$$

gdzie:

- $x_i(t)$ – stan zadania i w chwili t,
- s_i – rosnąca, ciągła funkcja szybkości zadania i, $s_i(0)=0$,
- $p_i(t)$ – ilość mocy pobierana przez zadanie i w chwili t.

Uwzględniając zależność szybkości wykonywania zadania od chwilowej ilości mocy mu przydzielonej, warunek (2) zakończenia zadania i w chwili C_i, którego rozpoczęcie nastąpiło w a_i, można teraz zapisać jako:

$$\int_{a_i}^{C_i} s_i(p_i(t))dt = w_i \tag{5}$$

Porównajmy krótko modele (1) i (4). Podstawowym mankamentem modelu (1) jest to, iż nie jest on wystarczająco ogólny, by mógł być wykorzystywany w wielu sytuacjach praktycznych. W modelu tym zakłada się na przykład konkretną postać funkcji poboru mocy. Chociaż jest to uzasadnione dla przypadku aktualnie dostępnych architektur systemów komputerowych (funkcja (1) dla technologii CMOS), przyszłe technologie mogą wymusić konieczność stosowania innych typów funkcji. W konsekwencji, model (1) nie nadaje się do analizy właściwości rozwiązań

optymalnych przy mniej restrykcyjnych założeniach nakładanych na postać funkcji poboru mocy.

Model (4) nie ogranicza się do jednej konkretnej postaci funkcji szybkości. Zakłada się w nim jedynie, że funkcja s_i musi być rosnąca i ciągła. Z tego powodu przy użyciu modelu (4) możliwe jest rozważanie dowolnej praktycznie uzasadnionej postaci funkcji szybkości.

Ponadto w modelu (1) zakłada się, że funkcja poboru mocy jest taka sama dla wszystkich zadań w systemie, a zadania różnią się jedynie rozmiarem. Jest to dopuszczalne w sytuacji, gdy każdy rozkaz procesora ma identyczną charakterystykę poboru mocy. Jak pokazano w rozdziale 2, istnieją jednak prototypy architektur systemów komputerowych, gdzie każdy rozkaz procesora może posiadać inną charakterystykę poboru mocy, a w konsekwencji również zużycia energii. Oznacza to, że każde zadanie obliczeniowe, złożone z różnych zestawów rozkazów procesora, musi mieć indywidualnie dobrane zarówno rozmiar jak i funkcję szybkości wykonywania, co możliwe jest przy zastosowaniu modelu (4). Dobór odpowiedniej funkcji szybkości dla zadania może być dokonany na drodze eksperymentalnej. Warto jednak zwrócić uwagę, że w przypadku traktowania pojedynczego zadania obliczeniowego jako sekwencji zadań cząstkowych ograniczonych kolejnościowo, rozmiar każdego z zadań cząstkowych może być utożsamiany np. z liczbą iteracji wykonywanych w ramach takiego zadania. Jest to łatwiejsze do oszacowania niż tradycyjna metoda określania rozmiaru zadania obliczeniowego za pomocą liczby niezbędnych do jego wykonania cykli CPU.

Jak łatwo zauważyć model (4) jest równoważny modelowi (1) jeśli funkcje szybkości zadań:

$$s_i(\cdot) = s(\cdot) = p^{-1}(\cdot) = p_i^{1/\alpha}, \;\; i = 1,2,\dots,n; \; \alpha > 1 \tag{6}$$

Ponadto, odwrotna niż w klasycznym modelu zależność między mocą a szybkością jest w modelu (4) uzasadniona występowaniem naturalnego ograniczenia na dostępną moc. Uwzględnienie limitu mocy jest oczywiste, gdyż maksymalny pobór mocy to jeden z parametrów charakteryzujących każdy procesor. Ponadto limit taki pozwala w naturalny sposób zabezpieczyć procesor wielordzeniowy przed ryzykiem przegrzania. Jeśli więc przez P ($P > 0$) oznaczymy całkowitą dostępną moc, sumaryczny pobór mocy przez n zadań w chwili t powinien spełniać następujące ograniczenie:

$$\sum_{i=1}^{n} p_i(t) \leq P \tag{7}$$

Podobny efekt, jak ten reprezentowany przez ograniczenie (7), w modelu (1) próbuje się osiągnąć przez wprowadzenie ograniczenia na maksymalną szybkość każdego rdzenia. Jak łatwo zauważyć nie jest to równoważne podejściu, w którym cała dostępna moc może być swobodnie rozdysponowywana pomiędzy równolegle wykonywane zadania.

Warto w tym miejscu podkreślić, że dynamiczny model zadań, dzięki swej ogólności i uniwersalności, w pełni nadaje się również do wykorzystania do modelowania problemów szeregowania w dużych centrach obliczeniowych wykorzystujących wirtualizację własnych zasobów komputerowych.

Na bazie modelu (4) sformułowano wiele praktycznych problemów szeregowania z różnymi kryteriami oceny uszeregowań. Problemy te były rozważane w licznych pracach naukowych (zob. np. [7] jako przegląd).

5. Podsumowanie

W niniejszej pracy porównano dwa znane z literatury modele wykonywania zadań obliczeniowych w energooszczędnych systemach komputerowych. Mimo że model (1) jest częściej wykorzystywany w pracach dotyczących energooszczędnego wykorzystywania komputerów, model (4) pozwala efektywniej dowieść wielu nietrywialnych własności uszeregowań optymalnych. Model (4), jako ogólniejszy, daje również możliwość modelowania najnowszych architektur procesorów o zmiennej szybkości wykonywania zadań. Dalsze kierunki prac obejmować mogą przypadki takich systemów komputerowych, które wymagają użycia w modelu (4) nietypowych funkcji szybkości.

Podziękowanie

Niniejsza praca jest częścią projektu sfinansowanego ze środków Narodowego Centrum Nauki, przyznanych na podstawie decyzji nr DEC-2013/08/A/ST6/00296.

LITERATURA

1. Boyer F.R., Epassa H.G., Savaria Y.: Embedded power-aware cycle by cycle variable speed processor. Computers and Digital Techniques, IEE Proceedings, 153 (4), 2006, p. 283-290.
2. Burd T.D., Pering T., Stratakos A., Brodersen R.W.: A dynamic voltage scaled microprocessor system. IEEE Journal of Solid-State Circuits, 35 (11), 2000, p. 1571-1580.
3. Burkov V.N.: Optimal project control. Prace IV Kongresu IFAC, tom 35, Warszawa, 1969, s. 46-57.
4. Kalisz J.: Podstawy elektroniki cyfrowej. Wydawnictwa Komunikacji i Łączności, Warszawa, 2002.
5. Różycki R.: Szeregowanie zadań obliczeniowych z uwzględnieniem ograniczeń energetycznych (rozprawa habilitacyjna). Wydawnictwo Nakom, Seria: Poznan Monographs in Computing and Its Applications, Nr 15, Poznań, 2013.
6. Węglarz J.: Time-optimal control of resource allocation in a complex of operations framework. IEEE Transactions on Systems Man and Cybernetics SMC, 6 (11), 1976, p. 783-788.
7. Węglarz J., Józefowska J., Mika M., Waligóra G.: Project scheduling with finite or infinite number of activity processing modes – A survey. European Journal of Operational Research, 208 (3), 2011, p. 177-205.