Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej Nr 22

XVI Seminarium
ZASTOSOWANIE KOMPUTERÓW W NAUCE I TECHNICE' 2006
Oddział Gdański PTETiS
Referat nr 8

FORMALIZM I METODY SZEREGOWANIA ZADAŃ DLA POTRZEB REDUKCJI POBORU MOCY CYFROWYCH UKŁADÓW CMOS

Krzysztof GIARO [1], Władysław SZCZEŚNIAK [2]

1. Politechnika Gdańska, ul. G. Narutowicza 11/12, 80-952 Gdańsk
   tel: (058) 347 24 28 fax: (058) 341 61 32 e-mail: giaro@eti.pg.gda.pl
2. Politechnika Gdańska, ul. G. Narutowicza 11/12, 80-952 Gdańsk
   tel: (058) 347 21 78 fax: (058) 347 20 90 e-mail: wlad@eti.pg.gda.pl

W pracy przedstawiono związki pomiędzy modelami formalnymi stosowanymi w klasycznym szeregowaniu zadań a metodami wykorzystywanymi w syntezie wysokiego poziomu układów cyfrowych CMOS. Zagadnienia optymalizacyjne pojawiające się w obu tych problemach mogą być w pewnym sensie transformowalne. Pozwala to na przenoszenie wybranych metod rozwiązań z jednego problemu do drugiego.

1. WPROWADZENIE

Szeregowanie zadań to jeden z głównych nurtów dziedziny zwanej badaniami operacyjnymi. Dotyczy on problemów związanych z harmonogramowaniem produkcji przemysłowej, planowaniem projektów i organizacją pracy. Od pewnego czasu w związku z szybkim rozwojem technologii komputerowych, pojawiają się też nowe zagadnienia, dotyczące przetwarzania procesów w wielozadaniowych systemach operacyjnych czy organizacji obliczeń rozproszonych. Do dziś w ramach tej dyscypliny zdefiniowano tysiące rozmaitych typów modeli związanych z konkretnymi zagadnieniami praktycznymi, a jej głównym celem jest opracowanie efektywnych algorytmów umożliwiających tworzenie harmonogramów optymalnych lub suboptymalnych w sensie określonego kryterium ich kosztu.

Instancja problemu *szeregowania na maszynach równoległych* składa się ze skończonego zbioru *zadań* $T=\{T_1,...,T_n\}$, które mają być wykonane przy użyciu *maszyn* (zwanych też *procesorami*), tworzących zbiór $M=\{M_1,...,M_m\}$ w taki sposób, by zredukować ustaloną funkcję celu dla określonego kryterium optymalizacyjnego zwanego *kosztem harmonogramu*. Przez *uszeregowanie* lub *harmonogram* rozumiemy przyporządkowanie procesorów do zadań w czasie, spełniające odpowiednie kryteria poprawności. Przyjmujemy, że mamy do dyspozycji dodatnią półoś rzeczywistą czasu podzieloną na kolejne jednostki numero-

Recenzent: Prof. dr hab. inż. Marek Kubale - Wydział Elektroniki, Telekomunikacji i Informatyki Politechniki Gdańskiej

wane liczbami naturalnymi. Standardowo zakłada się, że każde zadanie musi być wykonywane bez przerw w kolejnych chwilach (*niepodzielność* zadań jest cechą domyślną lecz nieobligatoryjną – w literaturze [2] badane są również systemy zadań podzielnych), maszyna zaś może w danym momencie pracować nad co najwyżej jednym zadaniem. Przykładowo w modelu maszyn *identycznych* czas wykonania dowolnego zadania T_i, i=1,...,n jest z góry znany i wynosi p_i, zaś najprostszym kryterium kosztu jest długość harmonogramu $C_{\max}=\max_{i=1,...,n} C_i$, gdzie C_i (ang. *completion time*) to chwila zakończenia zadania T_i. Poszukiwanie harmonogramu $C_{\max}$–optymalnego (czyli o najmniejszym możliwym $C_{\max}$) odpowiada więc takiemu rozplanowaniu zadań, w którym cała praca zakończy się możliwie najwcześniej. Innym, często stosowanym kryterium jest suma czasów wykonania wszystkich zadań ΣC_i, lub związany z nią średni czas przepływu zadania przez system.

Model maszyn identycznych jest oczywiście przypadkiem skrajnie uproszczonym, teoria szeregowania wprowadza wiele jego rozmaitych uogólnień [2]. Przykładowo czas wykonywania zadania może zależeć od przetwarzającego je procesora (tzw. maszyny równoległe *jednorodne* i *dowolne*). Często w zbiorze T wprowadza się relację *zależności kolejnościowych* (ang. *precedence constranis*), będącej częściowym porządkiem (relacja przeciwzwrotna i przechodnia) określającym dla każdego zadania T_i jego *poprzedników*, tj. zadania, które muszą zakończyć się przed rozpoczęciem T_i. Rozpatruje się też inne modyfikacje np. dodatkowe zasoby lub zadania *wieloprocesorowe*. Kompleksowy przegląd modeli i metod stosowanych w szeregowaniu zadań można znaleźć w pracach [1] i [2].

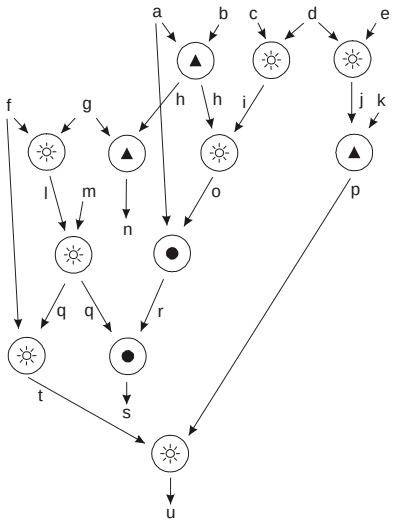
Jedną z głównych barier ograniczających obszar zastosowań klasycznych algorytmów szeregowania, (np. dla potrzeb planowania procesów w systemach komputerowych) jest determinizm badanych modeli. Przyjmuje się, że wszystkie parametry instancji problemu tj. zarówno zadań, jak i maszyn znane są przed rozpoczęciem pracy, stanowiąc dane wejściowe dla algorytmu planującego. Jest to jedna z przesłanek kierujących uwagę badacza w stronę zagadnień projektowania cyfrowych układów elektronicznych CMOS z redukcją mocy pobieranej ze źródła zasilającego, w których determinizm jest założeniem najzupełniej naturalnym. Projektant bowiem z góry wie, jaki proces przetwarzania danych musi wykonywać określony blok systemu/jednostka funkcjonalna, zaś od jakości uzyskanego projektu zależy zarówno koszt wytworzenia układu jak i jego parametry eksploatacyjne np. pobór mocy ze źródła zasilającego.

2. MODEL FORMALNY ZAGADNIENIA PROJEKTOWANIA UKŁADU

Zadanie syntezy wysokiego poziomu układu cyfrowego na ogół przedstawiane jest, przy pomocy grafu przepływu danych *DFG*(*V*,*E*). Jest to digraf acykliczny o wierzchołkach odpowiadających pojedynczym operacjom realizującym elementarne funktory, łuki jego zaś przedstawiają kierunek transmisji danych od wejść do wyjścia. Rys. 1 przedstawia przykład grafu *DFG* dla pewnego zadania obliczeniowego złożonego z 12 operacji realizowanych przez trzy różne funktory.

Projektant ma za zadanie przypisanie operacji do realizujących je jednostek funkcjonalnych w kolejnych cyklach zegarowych. Pożądane jest przy tym rozplanowanie pracy jednostek funkcjonalnych w możliwie najkrótszym odcinku odpowiadającym czasowi wykonywania przez układ wymaganego zadania.

MOST WIEDZY Pobrano z mostwiedzy.pl



Rys. 1. Przykładowy graf przepływu danych *DFG* [5]

Celem tego podrozdziału jest wprowadzenie modelu formalnego pozwalającego na opis zadania stojącego przed projektantem układu w ramach terminologii przyjętej w szeregowaniu zadań. Oczywistym jest, że poszczególne operacje interpretowane będą jako zadania do wykonania, zaś graf przepływu danych ściśle odpowiada częściowemu porządkowi określającemu zależności kolejnościowe pomiędzy zadaniami. I tak, jednostki funkcjonalne odpowiadają procesorom, a czas mierzy się pojedynczymi cyklami zegara. Wreszcie - odcinek czasu, w którym gotowy układ zakończy pracę ściśle odpowiada klasycznemu kryterium C_{max} dla końcowego harmonogramu. W tym jednak miejscu podobieństwa kończą się, bowiem zbiór problemów, przed którymi staje projektant jest szerszy niż w klasycznym szeregowaniu na maszynach równoległych.

Podstawową różnicą jest odrzucenie założenia o identyczności maszyn. Procesory realizują przypisane sobie funktory, podobnie funktor jest parametrem każdego z zadań, co narzuca oczywiste ograniczenia na to jakie zadanie można wykonać na jakiej maszynie. Dalej, poza kryteriami optymalizacyjnymi opartymi na harmonogramie (jak np. C_{max}) pojawiają się inne, konieczne do uwzględnienia koszty. Przesłanki zarówno ekonomiczne jak i konstrukcyjne czynią cechą pożądaną prostotę układu, co oznacza ograniczenie liczby używanych maszyn (tj. jednostek funkcjonalnych) poszczególnych typów. Różni się to zasadniczo od szeregowania klasycznego, gdzie zbiory zasobów (w tym procesorów) nie podlegają optymalizacji, lecz jako parametr instancji problemu są z góry dane. Wreszcie procesory dedykowane do wykonania jednego funktora w ramach tego samego układu, mogą być realizowane w rozmaitych technologiach, co wpływa zarówno na koszt wykonania układu jak i przede wszystkim jego parametry eksploatacyjne. Przykładowo często jest do dyspozycji kilka możliwych technik wytworzenia jednostki funkcjonalnej, różnią się zaś one szybkością działania oraz mocą pobieraną ze źródła zasilającego (zwykle rosnącą wraz ze wzrostem szybkości działania). Fakty te skłaniają do rozszerzenia klasycznego modelu formalnego zestawu zadań o nowe elementy, charakterystyczne dla zagadnień projektowania układów.

MOST WIEDZY Pobrano z mostwiedzy.pl

Definicja 1. Instancją zagadnienia *Projektowania Układu Cyfrowego CMOS* (*PUC*) jest następujący zestaw parametrów:

- skończone *zbiory zadań* $T=\{T_1,...,T_n\}$ oraz *funktorów* $S=\{s_1,...,s_l\}$,
- funkcje *typów zadań* $ts:T\to S$ oraz *różnorodności typów maszyn* $ms:S\to N$,
- funkcja *czasu wykonania* $p:S\times\{1,...,\max ms\}\to N$,
- relacja częściowego porządku *zależności kolejnościowej* $\prec$ określonej na zbiorze T,
- opcjonalnie: zbiór parametrów naturalnych m, $m(s)$ i $m(s,i)$ dla $s\in S$, $i\in\{1,...,ms(s)\}$ oraz funkcja *kosztu maszyn* $Cost:S\times\{1,...,\max ms\}\to N$.

Zadania zbioru T reprezentują operacje odpowiadające wierzchołkom grafu *DFG* danych reprezentowanego tutaj relacją porządku $\prec$. Każde z zadań realizuje pewien przypisany mu funktor, informację o nim zawiera funkcja *ts*. Niekiedy warto przyjąć upraszczające oznaczenie $s_i=ts(T_i)$ określające typ zadania T_i. Każdy z funktorów $s\in S$ można wykonywać na maszynach (jednostkach funkcjonalnych) rozmaitych typów różniących się parametrami użytkowymi – te rozróżnialne rodzaje procesorów numerujemy kolejnymi liczbami naturalnymi od 1 do $ms(s)$. Funkcja *ms* podaje zatem liczbę dostępnych technologii, w ramach których można przygotować maszynę dla funktora s. W szczególności od tegoż typu o numerze $j\in\{1,...,ms(s)\}$ zależy czas wykonywania zadania typu $s\in S$ równy $p(s,j)$. Jest on zatem nie tyle parametrem zadania, co rodzaju maszyny, na której zadanie będzie uruchamiane. Klasyczna funkcja czasu wykonania zadania $p:T\to N$ oraz jego standardowe oznaczenie $p_i=p(T_i)$ dla zadania T_i w zdefiniowanym modelu będą miały sens jedynie w szczególnym przypadku, kiedy to wszystkie funktory mogą być realizowane tylko w jeden sposób (czyli $ms\equiv 1$). Wówczas faktycznie można przyjąć upraszczającą notację $p(T_i)=p(ts(T_i),1)$. Projektant musi liczyć się ponadto z możliwymi ograniczeniami zasobowymi: maksymalną łączną liczbą maszyn m, maszyn $m(s)$ realizujących konkretne $s\in S$ oraz liczbą $m(s,i)$ tychże maszyn określonego typu $i\in\{1,...,ms(s)\}$. Dodatkowo funkcja *Cost* opisuje koszt (wykonania lub eksploatacji) wynikający z wprowadzania do układu procesorów różnych rodzajów – parametr ten może być np. związany z poborem mocy ze źródła.

Definicja 2. *Rozwiązaniem problemu PUC* lub *uszeregowaniem* dla instancji spełniającej warunki poprzedniej definicji nazywamy następujący zestaw parametrów:

1. skończony *zbiór maszyn* $M=\{M_1,...,M_m\}$ i jego rozbicie (tj. przedstawienie w postaci sumy parami rozłącznych podzbiorów) $M=\bigcup_{s\in S} M(s)$,
2. dla każdego $s\in S$ funkcje *typów maszyn* r_s: $M(s)\to\{1,...,ms(s)\}$ (ich sumę dla uproszczenia możemy oznaczyć przez rs: $M\to\{1,...,\max ms\}$),
3. funkcja *harmonogramu* $H:T\to M\times N$,

o ile spełnione są następujące warunki poprawności:

- każde zadanie wykonuje się na maszynie realizującej odpowiadający mu funktor

$$\forall_{T_i\in T}\ \forall_{s\in S}\ \ ts(T_i)=s \Rightarrow H(T_i)_1\in M(s)$$

- respektowane są ograniczenia kolejnościowe

$$\forall_{T_i,T_j\in T}\ T_i\prec T_j \Rightarrow H(T_i)_2+p(ts(T_i),rs(H(T_i)_1))\le H(T_j)_2,$$

- żaden procesor nie może równocześnie wykonywać dwóch różnych zadań

$$\forall_{T_i,T_j\in T}\ T_i\neq T_j \wedge H(T_i)_1=H(T_j)_1 \Rightarrow$$

$$\Rightarrow \{H(T_i)_2,...,H(T_i)_2+p(ts(T_i),rs(H(T_i)_1))-1\}\cap\{H(T_j)_2,...,H(T_j)_2+p(ts(T_j),rs(H(T_j)_1))-1\}=\varnothing,$$

MOST WIEDZY Pobrano z mostwiedzy.pl

- spełnione są ograniczenia zasobowe, tj. $|M|=m$, $|M(s)|=m(s)$ i $|r_s^{-1}(i)|=m(s,i)$ dla wszystkich $s\in S$ oraz $i\in\{1,...,ms(s)\}$, dla których instancja narzucała odpowiednie warunki.

W przeciwieństwie do klasycznego szeregowania zadań, rozwiązanie problemu PUC (poza przypisaniem zadań do maszyn w różnych momentach czasu) musi definiować zestaw wykorzystywanych procesorów oraz ich parametry. Są to: zbiór maszyn M, podzbiory $M(s)\subseteq M$ zawierające procesory realizujące funktory $s\in S$, zaś informacji o ich typach dostarczają funkcje r_s. Właściwego uszeregowania zadań z T na maszynach dokonuje funkcja H. Dla argumentu $T_i\in T$ pierwsza współrzędna $H(T_i)_1$ określa procesor, na którym zadanie ma być wykonane, gdy tymczasem $H(T_i)_2$ to numer jednostki czasu, w którym uruchamiane jest zadanie.

Definicja 3. Dla uszeregowania, będącego rozwiązaniem instancji problemu PUC, określa się funkcję *terminu zakończenia* zadań $C:T\to N$ przy pomocy formuły

$$\forall_{T_i\in T} C(T_i)=C_i=H(T_i)_2+p(ts(T_i),rs(H(T_i)_1))-1.$$

Definicja 4. Na zbiorze uszeregowań dla danej instancji problemu PUC określamy następujące funkcje kosztu:

- *długość uszeregowania* $C_{\max}=\max_{i=1,...,n} C_i$,
- łączną *liczbę maszyn* $|M|$, *maszyn realizujących funktor* s $|M(s)|$ oraz tychże maszyn określonego typu $|r_s^{-1}(i)|$ dla $s\in S$ oraz $i\in\{1,...,ms(s)\}$, o ile wielkości te nie były zdeterminowane parametrami instancji,
- *koszt układu* $CS=\Sigma_{s\in S, M_i\in M(S)} Cost(s,r_s(M_i))$, jeżeli instancja definiowała funkcję *Cost*.

Należy tu zaznaczyć, że klasyczne szeregowanie zadań używa jedynie pierwszego kryterium. Optymalizacja względem któregoś z parametrów wymienionych w punkcie drugim jest równoznaczna z poszukiwaniem „oszczędnej" konstrukcji o możliwie ograniczonej liczbie procesorów. Kosztowi układu można nadać zbliżony sens, jak również wielkość tą można wykorzystać do reprezentacji niekorzystnych parametrów eksploatacyjnych np. poboru mocy w trakcie pracy. Widać wyraźnie, że projektant układu cyfrowego staje w obliczu złożonego wielokryterialnego problemu optymalizacyjnego.

3. PROJEKTOWANIA UKŁADU CYFROWEGO A SZEREGOWANIE ZADAŃ – WZAJEMNE ZWIĄZKI

W podrozdziale tym przedstawiono dwa przykłady problemów optymalizacyjnych związanych z syntezą wysokiego poziomu układów cyfrowych, badanie których ułatwiło ich skojarzenie ze zbliżonymi zagadnieniami z zakresu szeregowania zadań. Na wstępie należy zwrócić uwagę, że w problemach klasy PUC naturalnym jest przyjęcie $C_{\max}$ jako podstawowego kryterium kosztu. Decyduje ono bowiem o szybkości realizacji zadania przez układ elektroniczny. Co więcej, optymalną długość harmonogramu $C_{\max}^*$ bez ograniczeń zasobowych (procesory/jednostki funkcjonalne są dostępne w dowolnych wymaganych ilościach) można łatwo uzyskać w czasie wielomianowym. W tym celu ograniczamy różnorodność maszyn (przyjmujemy $ms\equiv 1$), pozostawiając dla każdego $s\in S$ jedynie typ i o najkrótszym czasie wykonania $p(s,i)$. Wówczas funkcja czasu wykonywania dla zadań $p:T\to N$ jest jednoznacznie określona, a same zadania szeregujemy $C_{\max}$–optymalnie

MOST WIEDZY Pobrano z mostwiedzy.pl

z uwzględnieniem zależności kolejnościowych $\prec$ klasycznym algorytmem *ścieżki krytycznej* (w literaturze występującym także w jednej z wersji pod nazwą *ASAP* [6]). Jednak uwzględnienie kryteriów innych, niż $C_{\max}$, wymaga w dalszej kolejności wyjaśnienia w jakim stopniu można poprawić uzyskane uszeregowanie (a więc projekt układu) nie wydłużając czasu działania powyżej optymalnego $C_{\max}$?

3.1. Minimalizacja liczby procesorów

Zakładamy, że dla każdego funktora $s \in S$ instancja problemu zawiera tylko jeden typ maszyn (a zatem $ms \equiv 1$) bez ograniczeń zasobowych. Znając uszeregowanie $C_{\max}$–optymalne, celem tego etapu projektowania jest zmniejszenie liczby używanych procesorów bez wydłużania czasu działania. Łatwo zauważyć, że kryteria minimalizacji liczby maszyn $|M(s)|$ dla różnych funktorów s są ze sobą sprzeczne, a ich równoczesna optymalizacja może być niewykonalna.

W dalszym ciągu należy skupić się na minimalizacji liczby wszystkich maszyn $|M|$ jako drugim kryterium (podstawowym pozostaje $C_{\max}$). Okazuje się, że zagadnienie to jest trudne obliczeniowo nawet dla bardzo uproszczonych instancji. Przywołamy

Twierdzenie 1. *Problem odpowiedzi na pytanie, czy dla danego zbioru zadań jednostkowych i niepodzielnych* $\{T_1,\ldots T_n\}$ *powiązanych relacją zależności kolejnościowych i wykonywanych na identycznych maszynach* $M_1,\ldots,M_m$ *istnieje harmonogram o długości nie przekraczającej C* (*w notacji trójpolowej zagadnienie opisuje się jako* $P|p_j=1,prec|C_{\max}$,) *jest NP–trudny* [3]. ■

Twierdzenie 2. *Problem PUC z minimalizacją liczby maszyn jako drugim kryterium* (*po* $C_{\max}$) *jest NP–trudny nawet w przypadku, gdy istnieje tylko jeden funktor i jeden rodzaj maszyn dla niego* (*czyli* $|S|=1$ *i* $ms \equiv 1$), *a wszystkie czasy wykonywania są jednostkowe.*
Dowód. Przedstawiona zostanie wielomianowa redukcja wersji decyzyjnej problemu $P|p_j=1,\text{prec}|C_{\max}$ do zagadnienia minimalizacji liczby maszyn. Niech dany będzie zbiór zadań jednostkowych $\{T_1,\ldots,T_n\}$ z relacją częściowego porządku $\prec$, liczba procesorów m i limit czasu C. Bez zmniejszenia ogólności można przyjąć, że C jest nie mniejsze niż długość ścieżki krytycznej w digrafie zdefiniowanym przez relację $\prec$. Redukcja polega na wprowadzeniu nowych zadań jednostkowych Z_i, $i=1,\ldots,C$ z zależnościami kolejnościowymi w postaci łańcucha długości C. Zadania te pozostają czasowo niezależne od T_j, $j=1,\ldots,n$. Teraz długość ścieżki krytycznej wynosi C. Instancję problemu minimalizacji liczby maszyn otrzymujemy pytając, czy wszystkie te zadanie mogą być wykonane w czasie C na $m+1$ procesorach.

Jeżeli odpowiedź brzmi TAK, wówczas wszystkie zadania z łańcucha Z_i można umieścić na jednej maszynie, a pozostałe $T_1,\ldots T_n$ wykonują się na m procesorach w czasie nie przekraczającym C.

Z drugiej strony, jeśli istnieje odpowiedni harmonogram dla $T_1,\ldots T_n$, wówczas wprowadzamy dodatkowy procesor M_{m+1} dla zadań Z_i, $i=1,\ldots,C$ wykonywanych jedno po drugim. Długość takiego harmonogramu wynosi C. ■

Korzystając z rezultatów, znanych w teorii szeregowania zadań, dowiedliśmy że nie istnieje efektywny (o wielomianowej złożoności) algorytm wyznaczający uszeregowanie

MOST WIEDZY Pobrano z mostwiedzy.pl

o minimalnej liczbie maszyn bez wydłużania optymalnego czasu działania (o ile P≠NP). Próbując rozwiązać ten problem skazani jesteśmy zatem na algorytmy heurystyczne. Warto przywołać wyniki opisane w [5], gdzie zaproponowano procedurę, w której najpierw wyznaczano optymalne $C_{\max}^{*}$ metodą ścieżki krytycznej, a następnie w pętli dla kolejnych funktorów $s \in S$ zachłannie zmniejszano limit dostępnych maszyn $m(s)$ generując harmonogramy algorytmem tzw. szeregowania listowego, dopóki ich długość nie przekraczała $C_{\max}^{*}$.

3.2. Minimalizacja kosztu układu

Tym razem rozważamy instancję problemu zawierającą ograniczenia liczby maszyn poszczególnych typów $m(s)$ dla wszystkich funktorów $s \in S$ oraz pewne uszeregowanie o akceptowanej długości $C_{\max}=L$. Zadaniem naszym jest znalezienie nowego uszeregowania o nie większej długości niż L, respektującego te same ograniczenia zasobowe $m(s)$, z jednoczesną minimalizacją kosztu układu. Rzecz jasna, musi się to wiązać ze zmianą typów niektórych maszyn w zbiorach $M(s)$. W [6] zaproponowano przekonującą motywację praktyczną dla tego zagadnienia: należy zastąpić możliwie jak najwięcej jednostek funkcjonalnych danego układu wolniejszymi (a więc pobierającymi mniejszą moc ze źródła zasilającego) odpowiednikami bez wydłużania czasu działania całego układu.

Lemat 1. *Niech dany będzie harmonogram uzyskany procedurą szeregowania listowego o długości L dla zbioru zadań jednostkowych i niepodzielnych, powiązanych relacją zależności kolejnościowych i wykonywanych na identycznych maszynach. Problem odpowiedzi na pytanie, czy istnieje poprawny harmonogram długości nie przekraczającej L, w którym wszystkie procesory prócz jednego zastąpiono maszynami dwukrotnie wolniejszymi jest NP–zupełny.*

Dowód. Przedstawimy wielomianową redukcję z decyzyjnej wersji problemu $P|p_j=1,\text{prec}|C_{\max}$. Niech dane będą zadania jednostkowe $T_1,\ldots T_n$ z relacją zależności kolejnościowych $\prec$, liczba maszyn m oraz limit czasu C. Najpierw konstruujemy uszeregowanie S_A o pewnej długości C_A, używając dowolnej procedury szeregowania listowego. Jeśli $C_A \le C$, wówczas odpowiedź dla $P|p_j=1,\text{prec}|C_{\max}$ brzmi TAK, zaś jeżeli $C_A > 2C$, wówczas jest to NIE – korzystamy tu ze znanego twierdzenia orzekającego, że harmonogramy listowe są $(2-|M|^{-1})$–przybliżone dla kryterium $C_{\max}$ (por. [4]). Możemy zatem przyjąć $C_A/2 \le C < C_A$. Wprowadzamy łańcuch nowych zadań jednostkowych Z_i, $i=1,\ldots,2C$ czasowo niezależnych od T_j, $j=1,\ldots,n$. Następnie na podstawie S_A budujemy listowy harmonogram o długości $L=2C$ dodając nową maszynę M_{m+1} wykonującą kolejne zadania Z_i.

Jeżeli istnieje uszeregowanie dla $T_1,\ldots T_n$ na m maszynach o długości ograniczonej przez C, wówczas możemy dwukrotnie spowolnić te procesory, a następnie wprowadzić nową M_{m+1} o normalnej szybkości przetwarzającą kolejne Z_i. W ten sposób wszystkie zadania zostaną wykonane w czasie L.

Z drugiej strony, jeśli zadania T_i i Z_j mogą być wykonane w czasie $L=2C$ na m dwukrotnie wolniejszych maszynach i jednej normalnej M_{m+1}, wówczas łańcuch zadań Z_j, $j=1,\ldots,2C$ musi być przetwarzany przez M_{m+1}, czyli $T_1,\ldots T_n$ są uruchamiane wyłącznie na $M_1,\ldots,M_m$ i odpowiedź na $P|p_j=1,\text{prec}|C_{\max}$ brzmi TAK. ■

Jako wniosek otrzymujemy:

Twierdzenie 3. *Problem minimalizacji kosztu układu bez zwiększenia długości harmonogramu jest NP–trudny nawet w przypadku, gdy w instancji występuje tylko jeden funktor*

MOST WIEDZY Pobrano z mostwiedzy.pl

(*czyli* $|S|=1$) *oraz dwa typy maszyn: o szybkości normalnej i większym koszcie oraz dwukrotnie wolniejszych z mniejszym kosztem.* ■

Ponownie dowiedliśmy, że nie istnieje efektywna procedura dokonująca rzeczonej optymalizacji kosztu układu (o ile P≠NP). Można jednak stosować wielomianowe algorytmy heurystyczne. Takie podejście zaproponowano w [6], gdzie połączono schemat szeregowania listowego z pewną procedurą przenoszącą zadania na maszyny szybsze i spowalniającą procesory o małym obciążeniu.

4. PODSUMOWANIE

Sformułowanie zagadnień projektowania układów cyfrowych w ramach teorii szeregowania zadań przynosi różne korzyści, ułatwiając przy tym badanie problemów optymalizacyjnych, wobec których staje konstruktor układów. Wiele z tych zagadnień okazuje się być uogólnieniami modeli znanych od dawna w literaturze z zakresu badań operacyjnych. Pozwala to przenosić na grunt elektroniki rozmaite wyniki negatywne oraz wnioskować o NP–trudności napotykanych zagadnień. Można również żywić nadzieję na podobną przekładalność rezultatów pozytywnych tj. metod i algorytmów heurystycznych sprawdzonych przy konstrukcjach harmonogramów. Niektóre z nich winny pozwalać zaadaptować się do nowych zagadnień tworząc efektywne algorytmy wspomagające projektowanie układów cyfrowych CMOS. Próba takiego podejścia stanowi obiecujący obszar badawczy.

5. BIBLIOGRAFIA

1. Błażewicz J., Cellary W., Słowiński R., Węglarz J.: Badania operacyjne dla informatyków, Warszawa PWN 1983 ISBN 83-204-0519-X.
2. Błażewicz J., Dell'Olmo P., Drozdowski M., Speranza M.: Scheduling multiprocessor tasks on three dedicated processors, Inf. Process. Lett. 41, 1992, s. 275–280.
3. Garey M. R., Johnson D. S., Tarjan R. E., Yannakakis M.: Scheduling opposing forests, SIAM J. Algebraic Discrete Meth. 4, 1983, s.72–93.
4. Graham R. L.: Bounds for certain multiprocessing anomalies, Bell System Tech. J. 45, 1966, s. 1563–1581.
5. Szcześniak P., Szcześniak W.: Dobór optymalnej liczby jednostek funkcjonalnych dla realizacji syntezy wysokiego poziomu układów cyfrowych, Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Nr 21, Gdańsk 2005, s. 237-245, ISSN 1425-5766.
6. Szcześniak W., Voss B., Theisen M., Becker J., Glesner M.: Influence of high-level synthesis on average and peak temperatures of CMOS circuits, Microelectronics Journal, vol. 32, Oct. 2001, s. 855-862, ISSN 0026-2692.

PODZIĘKOWANIA

Niniejsza praca została w części sfinansowana przez Ministerstwo Nauki i Informatyzacji, grant nr 3T11B 015 27.

FORMALISM AND METHODS OF TASK SCHEDULING APPLIED TO POWER REDUCTION OF DIGITAL CMOS CIRCUITS

The paper presents relationship between formal models used in classic task scheduling and methods used in high level synthesis of digital CMOS circuits. The optimisation problems showing up in both of the problems can be transformed in both ways to some extent. The fact enables transition of solutions from one problem to another.

MOST WIEDZY Pobrano z mostwiedzy.pl