ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1981

Seria: INFORMATYKA z. 2 Nr kol. 683

Marek CHMURA

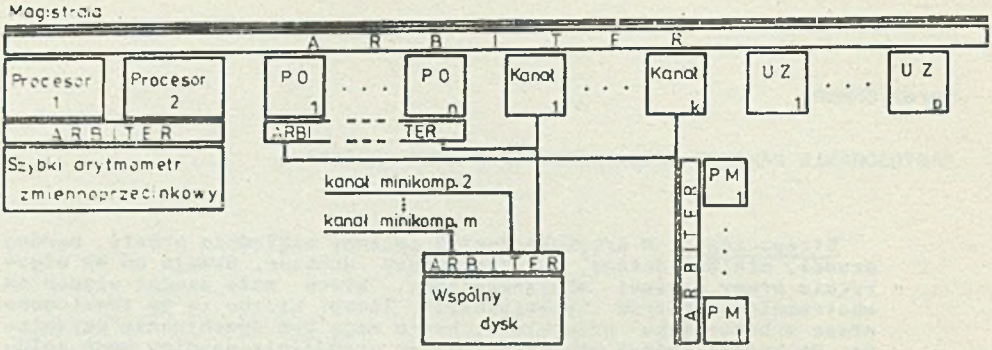
ZASTOSOWANIE PAMIĘCI SKOJARZENIOWEJ W ROLI ARBITRA

Streszczenie. W artykule został opisany względnie prosty, bardzo szybki, wielowejściowy, skojarzeniowy arbiter. Bazuje on na algorytmie pracy pamięci skojarzeniowej, która może szukać elementów ekstremalnych pośród zapamiętanych liczb. Liczby te są traktowane przez arbiter jako priorytety, które mogą być dynamicznie zmieniane. Po każdej jednej obsłudze arbiter przydziela wspólny zasób zgłoszonemu urządzeniu z największym priorytetem.

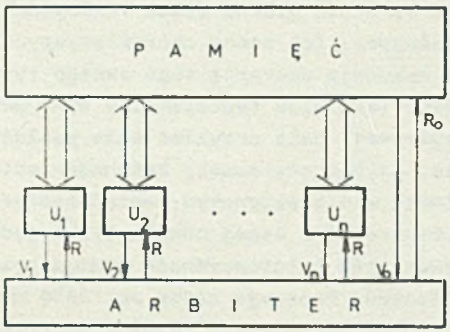
1. WSTĘP

W strukturze maszyny cyfrowej można wyróżnić bierne bloki funkcjonalne pełniące w stosunku do innych rolę usługową. Ich cechą charakterystyczną jest to, że dla każdego użytkownika wykonują operację tego samego typu. Jednocześnie mogą wykonywać tylko jedną. Taki blok funkcjonalny bywa określany niepodzielnym zasobem maszyny cyfrowej. Jako przykład może posłużyć: pamięć operacyjna, wspólna magistrala, szybki arytmometr zmiennoprzecinkowy, szybka mnożarka, wspólna w systemie wielomaszynowym pamięć masowa i inne. Z niepodzielnego zasobu może korzystać w danej chwili tylko jeden użytkownik. Ze względu na liczbę użytkowników i intensywność obsługi przed zasobami będą się tworzyły kolejki zgłoszeń. Powstaje zatem potrzeba wprowadzenia układu, który zarządzałby kolejką wg zadanego regulaminu. Układ pełniący taką funkcję jest nazywany układem przydzielającym niepodzielne zasoby, układem dostępu lub arbitrem. Oprócz wymienionych cech, arbiter powinien posiadać umiejętność rozwiązywania sytuacji konfliktowych, polegających na jednoczesnym zgłoszeniu się kilku użytkowników. Miejsca występowania w systemie obliczeniowym sytuacji konfliktowych wymagających użycia arbitra przedstawiono przykładowo na rys. 1.1. Rola arbitra w dostępie do zasobu polega na przyjmowaniu zgłoszeń V_i (rys. 1.2), umieszczaniu ich w kolejce, podejmowaniu decyzji zgodnie z zadanym regulaminem najczęściej w momentach zwalniania zasobu i wysyłaniu do urządzenia sygnału zezwolenia R_i. Nie pośredniczy on w wymianie informacji pomiędzy urządzeniem a zasobem. Urządzenie komunikuje się z arbitrem wyłącznie za pomocą przedstawionych sygnałów. Po otrzymaniu sygnału zezwolenia urządzenie przejmuje kontrolę nad interfejsem zasobu przez okres czasu narzucony bezpośrednio przez arbiter i uruchamia układy współpracy.

Efektywność pracy arbitra decyduje o stopniu wykorzystania niepodzielnych zasobów, co z kolei ma decydujący wpływ na moc obliczeniową maszyny



Rys. 1.1. Ogólna struktura minikomputera o architekturze opartej na wspólnej magistrali z zaznaczonymi, najważniejszymi miejscami występowania konfliktów



Rys. 1.2. Powiązanie arbitra z użytkownikami i zasobem

V_O - sygnał gotowości zasobu, R_O - sygnał przerwania obsługi (dla RPB - Regulamin z Priorytetami Bezwzględnymi)

i całego systemu. Od układu dostępu wymaga się by na drodze użytkownik - zasób wprowadzał jak najmniejsze opóźnienie związane z czasem podejmowania decyzji. Układy te buduje się zarówno jako asynchroniczne [1], [2], [3], [6] jak i synchronizowane [4] [5] [7]. Niezależnie od rozwiązań konstrukcyjnych zarządzają kolejką wg różnych regulaminów. Najczęściej stosowane to: RPW, RC, RN rzadziej RPB, RPD. W asynchronicznych arbitrach czas własny rozumiany jako czas podjęcia decyzji zależy często od priorytetu zgłoszenia. Zależność ta bywa najczęściej nieliniowa. Związana z nią jest kolejność przydzielania priorytetów użytkownikom uporządkowanym na podstawie intensywności strumieni zgłoszeń. Pociąga to za sobą odpowiednie zwiększenie czasu obsługi zgłoszeń użytkowników wolniejszych. To rozwiązanie spełnia oczekiwania, gdy z danego zasobu korzystają użytkownicy o silnie zróżnicowanych strumieniach zgłoszeń.

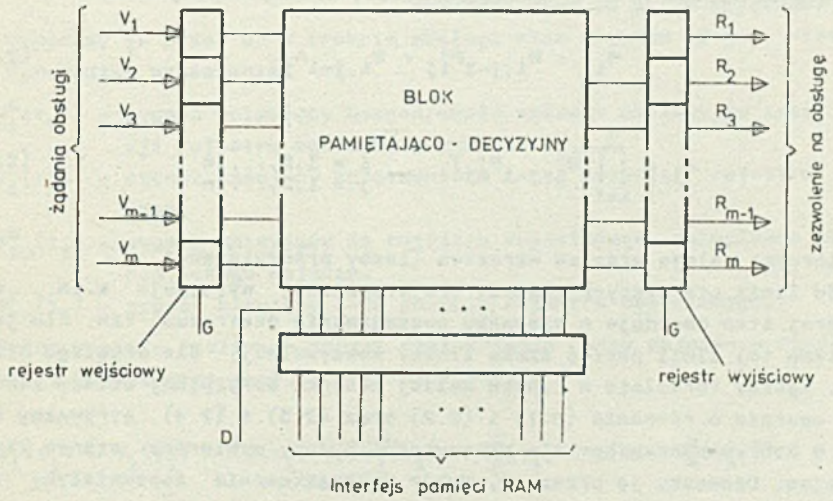
W systemie wieloprocesorowym zachodzi potrzeba obsługi kilku procesorów żądających dostępu do pamięci. Rozwiązanie przydziału oparte na przedstawionym typie asynchronicznego arbitra staje się niekorzystne. Arbiter spełniający to zadanie powinien się charakteryzować niezależnym od priorytetu użytkownika jak najmniejszym czasem decyzji. Priorytet powinien być

wykorzystywany tylko w celu rozwiązania sytuacji konfliktowych. Pożądaną cechą byłaby możliwość bieżącej zmiany struktury priorytetów.

Arbiter spełniający te wymagania jest przedmiotem niniejszego artykułu. Opracowany został w wersji asynchronicznej jako bardzo szybki arbiter oparty na oryginalnej konstrukcji pamięci skojarzeniowej [8].

2. SKOJARZENIOWY ARBITER

W pracy [8] przedstawiona została budowa pamięci skojarzeniowej przystosowanej do szukania elementów o wartościach ekstremalnych w zadanym przez stan rejestru wejściowego zbiorze komórek pamięci. Umieszczając w każdej inną co do wartości liczbę w rejestrze wyjściowym, otrzymamy dla dowolnego stanu rejestru adresowego odpowiedź w kodzie 1 z n. Ta właściwość predystynuje pamięć skojarzeniową do zastosowania w roli arbitra,



Rys. 2.1. Schemat blokowy arbitra wykorzystującego pamięć skojarzeniową

rys. 2.1. Opisany w [8] algorytm pracy elementarnej komórki pamięci związanej z jednym bitem jej słowa zwanej komórką decyzyjną, zezwala na szukanie liczby ekstremalnej w zbiorze liczb o różnych znakach. Stosując pamięć do budowy arbitra zauważmy, że liczby zapisane w komórkach będą pełniły rolę priorytetów. W związku z tym mogą to być liczby wyłącznie o tym samym znaku. Uprości to budowę komórki decyzyjnej. Przyjmujemy, że w dalszych rozważaniach będziemy korzystali z liczb dodatnich. Będziemy je traktować jak liczby bez znaku w wyniku włączenia go do modułu liczby.

Równanie opisujące algorytm działania komórki decyzyjnej będą miały dla powyższych założeń następującą postać:

$$W_{ij} = W_{i,j-1}P_{ij} \vee W_{i,j-1}A_j \tag{2.1}$$

$$A_j = \prod_{i=1}^{m} (W_{i,j-1}P_{ij})' \qquad \begin{matrix} i = 1,2,\ldots,m \\ j = 1,2,\ldots,n \end{matrix} \tag{2.2}$$

gdzie:

W_{ij} - stan sygnału analizującego j-ty bit i-tej komórki pamięci skojarzeniowej,

P_{ij} - wartość j-tego bitu i-tej komórki pamięci,

A_j - sygnał decyzyjny j-tej kolumny,

$(\cdot)'$ - negacja wyrażania logicznego,

m - pojemność pamięci w słowach,

n - długość słowa pamięci w bitach,

gdy priorytet rośnie wraz z wartością liczby priorytetowej oraz:

$$W_{ij} = W_{i,j-1}P'_{ij} \vee W_{i,j-1}A_j \tag{2.3}$$

$$A_j = \prod_{i=1}^{m} (W_{i,j-1}P'_{ij})' \qquad \begin{matrix} i = 1,2,\ldots,m \\ j = 1,2,\ldots,n \end{matrix} \tag{2.4}$$

gdy priorytet maleje wraz ze wzrostem liczby priorytetowej.

Wśród linii sterujących pamięci skojarzeniowej występuje m.in., taka, której stan decyduje o kierunku poszukiwania ekstremum, tzn. dla jednego stanu tej linii pamięć szuka liczby maksymalnej, dla drugiego minimalnej. Łącząc formalnie w ramach każdej komórki decyzyjnej układy zbudowane w oparciu o równania (2.1) i (2.2) oraz (2.3) i (2.4) otrzymamy arbiter, w którym uporządkowanie priorytetów byłoby wybierane stanem linii interfejsu. Oznaczmy ją przez D. Wybór uporządkowania zapewniałyby zależności:

$$W_{iO}^{max} = V_i^O D \tag{2.5}$$

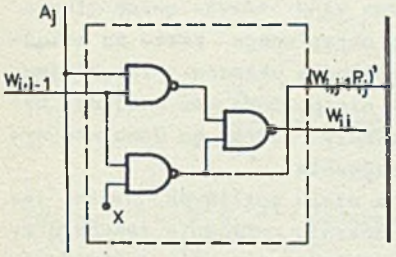
$$W_{iO}^{min} = V_i^O D, \tag{2.6}$$

gdzie:

W_{iO}^{max} - sygnał analizujący i-tego słowa szukający liczby maksymalnej,

V_i^O - sygnał bezpośrednio z i-tej pozycji rejestru adresowego,

W_{iO}^{min} - sygnał analizujący i-tego słowa szukający liczby minimalnej.



Rys. 2.2. Komórka decyzyjna arbitra dla $X = P_{ij}$ - umożliwia szukanie elementu maksymalnego, $X = \overline{P_{ij}}$ - umożliwia szukanie elementu minimalnego

Przyjmując jedną z dwóch możliwych wersji uporządkowania priorytetów, budowa komórki decyzyjnej uprości się do postaci przedstawionej na rys. 2.2. Wiąże się to z przyjęciem w równaniach (2.5) i (2.6) D = 1 lub D = 0.

Stosując pamięć skojarzeniową w roli arbitra należy nieco inaczej zorganizować jej rejestry. Rejestr adresowy będzie pełnił rolę rejestru zgłoszeń i powinien się charakteryzować możliwością wpisu asynchronicznego. Rejestr wyjściowy również powinien posiadać tę możliwość. Prawidłowa praca arbitra wymaga uzależnienia wpisu sygnałów do obu rejestrów od stanu zajętości zasobu. Oznaczmy go przez G. W trakcie obsługi stan sygnału G jest nieaktywny. Wprowadźmy oznaczenia:

$V_i^s(t_i)$ - sygnał wpisujący bezpośrednio żądanie obsługi do i-tej pozycji rejestru zgłoszeń w chwili t_i,

$V_i^r(t_i)$ - sygnał kasujący zgłoszenie w i-tej pozycji rejestru zgłoszeń,

$W_{in}^s(t_i)$ - sygnał wpisujący do rejestru wyjściowego zezwolenie obsługi dla i-tego wejścia,

$W_{in}^r(t_i)$ - sygnał kasujący i-tę pozycję rejestru wyjściowego.

Jedna z wielu możliwych wereji uzależnienia przy założeniu nieprzerywalności rozpoczętej obsługi jest następująca:

$$V_i^s(t_i) = V_i(t_i) \wedge (G(t_i) \wedge (V_1^{o'}(t_i) \wedge V_2^{o'}(t_i) \wedge \ldots \wedge V_m^{o'}(t_i))')' \qquad (2.7)$$

$$V_j^r(t_i) = (G(t_i) \wedge R_j(t_i))' \qquad (2.8)$$

$$W_{in}^s(t_i + \lambda) = W_{in}(t_i + \lambda_0 + \lambda_a)G(t_i + \lambda) \qquad (2.9)$$

$$W_{jn}^r(t_i + \lambda_0) = G(t_i + \lambda_0), \qquad (2.10)$$

gdzie:

t_i - określa moment przejścia sygnału G w sten aktywny,

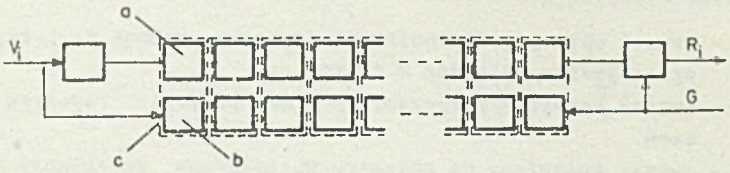
λ - opóźnienie wymagane w celu regeneracji stanu wyjściowego układów interfejsu zasobu lub wpisania poprawnego stanu do rejestru wyjściowego, powinno zachodzić $\lambda \geq \lambda_0 + \lambda_a$,

λ_a - czas pracy arbitra,

λ_0 - czas potrzebny na wyzerowanie i-tej pozycji w rejestrze zgłoszeń.

Niektóre zasoby maszyny cyfrowej wymagają określonego czasu po zakończeniu obsługi w celu regeneracji stanu wyjściowego układów interfejsu. Jeżeli czas λ_a będzie zbyt duży, co raczej nie powinno mieć miejsca, będą ponoszone określone straty, gdy będzie za mały można go dowolnie wydłużyć budując arbiter dla konkretnego zastosowania.

Przedstawione zabezpieczenie jest jednym z wielu możliwych. Efektem jego działania jest blokada wpisu zgłoszeń od momentu zwolnienia zasobu przy założeniu, że czekało co najmniej jedno zgłoszenie do momentu rozpoczęcia nowej obsługi. W czasie trwania obsługi jest blokowany wpis do rejestru wyjściowego, natomiast rejestr wejściowy może przyjmować zgłoszenia i tworzęca się kolejka może być porządkowana wg zadanego regulaminu. Możliwość realizacji różnych regulaminów wynika z dwuwarstwowej struktury pamięci a zatem i arbitra, rys. 2.3. O regulaminie będzie decydowała organizacja warstwy pamiętającej.



Rys. 2.3. Dwuwarstwowość struktury arbitra
a - komórka decyzyjna, b - komórka pamiętająca jeden bit, c - podstawowa komórka arbitra, G - sygnał gotowości zasobu

W różnych zastosowaniach może występić potrzeba zbudowania arbitra o czasie decyzji niezależnym od numeru zgłoszenia. W pracy [8] podana została zależność określająca szybkość działania warstwy decyzyjnej, czyli zbioru komórek decyzyjnych w odniesieniu do pojedynczego słowa. Definiuje się ją jako przedział czasu τ pomiędzy przejściem w stan aktywny sygnału W_{i0} a W_{in}.

$$\tau = \tau_d\left[n + i(\gamma - 1)\right], \tag{2.11}$$

gdzie:

τ_d - czas przejścia stanu aktywnego sygnału analizującego przez j-tą komórkę decyzyjną przy $P_{ij} = 1$ (rys. 2.2),

n - długość słowa w bitach,

i - ilość zer w słowie,

γ - iloraz maksymalnego do minimalnego czasu przejścia aktywnego stanu sygnału analizującego przez j-tą komórkę decyzyjną.
Prawidłowa praca pamięci wymaga by $\gamma \geq 2$.

Z zależności (2.11) wynika, że czas decyzji niezależny od numeru zgłoszenia można uzyskać przy spełnieniu warunku

$$\sum_{j=0}^{n-1} \alpha_{ij} = \sum_{j=0}^{n-1} \alpha_{kj} \quad \forall i,k \in \{1,2,\ldots,m\},\ i \neq k$$

gdzie $k_i = \sum_{j=0}^{n-1} \alpha_{ij} 2^j$ oznacza liczbę priorytetową i-tego wejścia. Narzuca to z kolei wybranie takiego n, by $C_n^i > m$, gdzie m oznacza przewidywaną liczbę użytkowników. W celu zwiększenia szybkości dążymy do minimalizacji i. Jeżeli $\gamma > 2$, to lepiej rozszerzyć liczbę priorytetową o jedną pozycję n := n + 1 a zmniejszyć i := i - 1. Należy jednak mieć na uwadze, że $C_n^i > C_{n+1}^{i-1}$ dla $n > 3i$. Można ten proces, który przyspieszy działanie arbitra kontynuować tak długo, jak długo będzie jeszcze zachodzić zależność $C_{n+k}^{i-k} > m$. Dla $\gamma = 2$ nie wystąpi przyspieszenie działanie układu w wyniku powyższych zabiegów.

W warstwie pamiętającej można wprowadzić dodatkową komórkę m+1. Jeżeli uwzględnimy ją w równaniach komórki decyzyjnej (2.1), (2.2) lub (2.3), (2.4), to jej zawartość będzie oznaczać dolny próg liczby priorytetowej. Wpisując do komórki m+1 liczbę 00...01 spowodujemy, że wyzerowanie dowolnej komórki od 1 do m pociągnie za sobą blokadę zgłoszeń danego wejścia. Inna co do wartości liczba wywoła blokadę wszystkich wejść o mniejszych od niej liczbach priorytetowych. Kombinacja samych jedynek, jeżeli nie występuje w roli liczby priorytetowej może służyć do blokady pracy arbitra. Jeżeli nie wprowadzimy powyższych zastrzeżeń, to liczba priorytetowa zero będzie oznaczać najniższy lub najwyższy priorytet w zależności od uporządkowania. Może to mieć znaczenie wówczas, gdy bardzo zależy nam na dużej szybkości obsługi wejść o wysokich priorytetach a liczy się wtedy każda pozycja binarna liczby priorytetowej, szczególnie gdy układ ma być wykonany z układów scalonych SSI i MSI np. serii SN74.

3. ZAKOŃCZENIE

Przedstawiony arbiter cechuje bardzo duża szybkość działania w odniesieniu do funkcji jakie przy tym spełnia. Umożliwia on sprzętową realizację różnych regulaminów obsługi, co wymaga wprawdzie odpowiedniej budowy warstwy pamiętającej ale dla niektórych regulaminów np. RPD - Regulaminu z Priorytetami Dynamicznymi, stwarza jedyną realną możliwość zastosowania w praktyce.

Należy podkreślić jedną z większych zalet tak zbudowanego arbitra. Polega ona na możliwości zmiany przyporządkowania priorytetów poszczególnym wejściom. Można to wykonać poza czasem podejmowania decyzji w dowolnej

chwili. Skojarzeniowy arbiter wyposażony w dodatkowy układ szybkiej obsługi warstwy pamiętającej, może w skrajnym przypadku w czasie jednej obsługi zmienić priorytety wszystkim sygnałom wejściowym.

W przytoczonych rozważaniach zakładano realizację arbitra w oparciu o podstawowe funktory wykonane technikę układów scalonych. Prosta budowa arbitra, szczególnie dla RPW - Regulaminu z Priorytetami Względnymi, pozwala na wykonanie w tej technice efektywnie pracującego układu o 8 a nawet 16 wejściach. Ze względu na powiązania pomiędzy komórkami decyzyjnymi, układami pomocniczymi i rejestrami, arbiter powinien być wykonany w formie zwartego układu. Takie rozwiązanie nie będzie uciążliwe nawet przy większej liczbie użytkowników, do każdego należy bowiem doprowadzić tylko dwa przewody. Najczęściej w zastosowaniach występuje potrzeba zarządzania kolejką wg RPW. W związku z tym przy ustalonej strukturze priorytetów można pominąć warstwę pamiętającą przez "zaszycie" liczb priorytetowych.

Do wad skojarzeniowego arbitra należy zaliczyć szybki wzrost liczby funktorów z ilością obsługiwanych wejść.

Opisany układ cechuje jednak duża uniwersalność predystynująca go do wielu praktycznych zastosowań.

LITERATURA

[1] W.W. Plummer: Asynchronous Arbiters, IEEE Transactions on Computers, vol. c-21, No - 1, 1972, pp. 37-42.

[2] R.C. Pearce, J.A. Field and W.D. Little: Asynchronous Arbiter Module, IEEE Transactions on Computers, (Corresp.), vol. c-24, No - 10, 1975, pp. 931-932.

[3] K. Soe Hojberg: An Asynchronous Arbiter Resolves Resource Allocation Conflicts On a Random Priority Basis, Computer Design, vol. 16, No - 8, 1977, pp. 120-123.

[4] K. Soe Hojberg: One-Step Programmable Arbiters for Multiprocessors, Computer Design, vol. 17, No - 4, 1978, pp. 154-158.

[5] K. Soe Hojberg: Queue Handling Arbiter Solves Shared Resource Conflicts, Computer Design, vol. 18, 1979, pp. 129-135.

[6] Minikomputer MERA 400. Interfejs., DTR, t. IV, cz. 1, 1978.

[7] S.J. Durham: Fast LSI Arbiters Supervise Priorities for Busaccess in Multiprocessor Systems, Electronic Design, No - 11, 1979, pp. 128-133.

[8] M. Chmura: Pamięć skojarzeniowa, Zeszyty Naukowe Politechniki Śląskiej, seria Informatyka (w druku).

Recenzent
Doc. dr hab. Wiesław Traczyk

Wpłynęło do Redakcji 2.04.1980 r.

ПРИМЕНЕНИЕ АССОЦИАТИВНОГО ЗАПОМИНАЮЩЕГО УСТРОЙСТВА
В КАЧЕСТВЕ АРБИТРА

Р е з ю м е

В статье описан довольно простой, очень быстродействующий, многовводной, ассоциативный арбитер. Он базирует на алгоритме работы ассоциативного запоминающего устройства, которое может искать экстремальные элементы среди сохраненных в памяти чисел. Эти числа арбитер считает приоритетами, которые могут быть динамически изменяемыми. После каждого одного обслуживания арбитер наделяет общим фондом предъявленное с наибольшим приоритетом устройство.

ASSOCIATIVE MEMORY APPLICATION AS AN ARBITER

S u m m a r y

In the paper a relatively simple high speed multichannel associative arbiter has been described. The arbiter bases on the algorithm on an associative memory, which can search for the extremal element among stored numbers. The numbers are used as a priority indices and can be changed dynamically. After each service the arbiter allocates the shared resource to the requesting device with largest priority by activating its grant line and simultanously it resets corresponding grant line of the resource which recently has been served.