3 1989

informatyka

Prof. Herbert Simon o rozwiązywaniu problemów

Stacja robocza Sun-3

dBase III Plus w sieciach lokalnych

WYDAWNICTWO NOT SIGMA

Cena 300 zł

informatyka

Nr 3

Miesięcznik Rok XXIV
Marzec 1989

Organ Komitetu
Naukowo-Technicznego NOT
ds. Informatyki

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 01-517 Warszawa, ul. Mickiewicza 18 m. 17, tel. 39-14-34

RSW „PRASA-KSIĄŻKA-RUCH"
PRASOWE ZAKŁADY GRAFICZNE
ul. Dworcowa 13, 85-950 BYDGOSZCZ
Zam. 323/89 E-3
Obj. 4,0 ark. druk. Nakład 8950 egz.

ISSN 0542-9951. INDEKS 36124

Cena egzemplarza 300 zł
Prenumerata roczna 3600 zł

WYDAWNICTWO
NACZELNA ORGANIZACJA TECHNICZNA
SIGMA
CZASOPISM I KSIĄŻEK TECHNICZNYCH

00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE:

W NAJBLIŻSZYCH NUMERACH:

- Wiesław Nowiński charakteryzuje transformacyjne metody rekonstruowania obrazów oraz oparte na tych metodach algorytmy i sposób ich implementacji.
- Marek Pawłowski dokonuje przeglądu układów szybkiego mnożenia.
- Kazimierz Koziarski opisuje wersję języka Cobol stosowaną na komputerach klasy IBM PC oraz metodę i technikę przenoszenia oprogramowania z Odry i Riada na PC.
- Michał Hornowski przedstawia metodę instalowania bardzo dużych programów napisanych w językach wysokiego poziomu.
- Jan Bielecki podaje przykłady konstrukcji oprogramowania przenośnego w języku Turbo C.

WARUNKI PRENUMERATY

Prenumeratorzy zbiorowi – jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłat wyłącznie na blankiecie „wpłata-zamówienie" (jest to „polecenie przelewu" rozszerzone dla potrzeb Wydawnictwa o część dotyczącą zamówienia). Blankiety te będą dostarczane dotychczasowym prenumeratorom przez Zakład Kolportażu. Nowi prenumeratorzy otrzymają je po zgłoszeniu zapotrzebowania (pisemnie lub telefonicznie) w Zakładzie Kolportażu.

Prenumeratorzy indywidualni – osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty. Wpłacać należy na konto: Państwowy Bank Kredytowy III/O Warszawa nr 370015-7490-139-11.

Prenumerata ulgowa – przysługuje wyłącznie osobom fizycznym – członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły. Sposób zamawiania prenumeraty ulgowej jest taki sam jak prenumeraty indywidualnej. W prenumeracie ulgowej można zamówić tylko po jednym egzemplarzu każdego czasopisma. Uwaga! Miesięcznik „Aura" może być zamawiany w prenumeracie ulgowej również przez uczniów szkół ogólnokształcących.

Prenumeratę ze zleceniem wysyłki za granicę – zamawia się tak, jak prenumeratę indywidualną. Dodatkowo należy podać na blankiecie wpłaty nazwisko i dokładny adres odbiorcy. Cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa.

Wpłaty na prenumeratę przyjmowane są w terminach: do 10 listopada na każdy kwartał, I i II półrocze oraz cały rok następny; do 28 lutego na II, III i IV kwartał oraz II półrocze; do 31 maja na III i IV kwartał oraz II półrocze; do 31 sierpnia na IV kwartał. Zmiany w prenumeracie można zgłaszać pisemnie tylko w wyżej wymienionych terminach.

Informacji o prenumeracie udziela – Zakład Kolportażu Wydawnictwa NOT SIGMA (ul. Bartycka 20, 00-716 Warszawa), skr. poczt. 1004, 00-950 Warszawa, tel. 40-00-21, wewn. 248, 249, 293, 297, 299 lub 40-30-86 i 40-35-89.

Egzemplarze archiwalne czasopism – można nabyć za gotówkę w Klubie Prasy Technicznej, Warszawa ul. Mazowiecka 12 (tel. 26-80-16), lub zamówić pisemnie. Zamówienia na egzemplarze archiwalne czasopism przyjmuje: Zakład Kolportażu, Dział Handlowy, 00-950 Warszawa, skr. poczt. 1004 (tel. 40-37-31), na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

Cena prenumeraty

miesięczna		kwartalna		półroczna		roczna	
normalna	ulgowa	normalna	ulgowa	normalna	ulgowa	normalna	ulgowa
300 zł	60 zł	900 zł	180 zł	1800 zł	360 zł	3600 zł	720 zł

HERBERT A. SIMON
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, stan Pensylwania
USA

Procesy przeszukiwania i wnioskowania w rozwiązywaniu problemów (1)

W trakcie intensywnych badań nad sztuczną inteligencją ukształtowało się kilka różnych sposobów widzenia i opisu procesu rozwiązywania problemów. Jedno z podejść traktuje rozwiązywanie problemów jako proces poszukiwania rozwiązań. Zakłada się istnienie pewnej przestrzeni, w której znajdują się ukryte skarby. Tworzy się symboliczne struktury (węzły), które modelują tę przestrzeń, oraz operatory przejścia, które zmieniają te symboliczne struktury, przenosząc nas od jednego węzła do innego. Rozwiązywanie problemów polega w tym wypadku na (selektywnym) przeszukiwaniu modelu przestrzeni rozwiązań i przechodzeniu od węzła do węzła, aż ukryty skarb zostanie znaleziony.

Drugie podejście traktuje rozwiązywanie problemów jako proces wnioskowania. Zakłada się istnienie pewnego systemu logiki, który umożliwia wydedukowanie nowych zdań z aksjomatów i ze zdań wydedukowanych uprzednio. Przedstawia się problem za pomocą zbioru aksjomatów w formalnym języku logiki. Rozwiązywanie problemów polega w tym wypadku na stopniowym gromadzeniu informacji (zdań) przez wnioskowanie, aż rozwiązanie danego problemu zostanie znalezione.

Trzecie z możliwych podejść traktuje rozwiązywanie problemów jako spełnienie ograniczeń. Zakłada się istnienie zbioru obiektów i różnych podzbiorów zdefiniowanych przez ograniczenia spełniane przez te obiekty. Rozwiązanie problemu polega tutaj na zawężeniu pierwotnego zbioru do pewnego podzbioru obiektów lub do konkretnego obiektu spełniającego wszystkie ograniczenia.

Podejścia te nie są w żadnym wypadku wzajemnie rozłączne. Ten sam algorytm rozwiązywania problemów można uważać raz jako poszukiwanie rozwiązania, raz – jako wnioskowanie, innym razem – jako spełnianie ograniczeń. Rozważmy dla przykładu prosty program dowodzenia twierdzeń, który działa „do przodu" na podstawie określonego zbioru aksjomatów stosując reguły wnioskowania do dostępnych aksjomatów i wyznaczając nowe wyrażenia, które mogą być dodane do zbioru aksjomatów. Problem jest rozwiązany, jeśli program znajdzie właściwą ścieżkę prowadzącą do żądanego twierdzenia. Z pewnością jest to więc algorytm poszukiwania rozwiązań.

Jednocześnie program dowodzący w każdym kroku procesu przeszukiwania generuje nowe zdania, które wynikają logicznie z aksjomatów. Stopniowo gromadzi on coraz większy zbiór wydedukowanych zdań, a więc jest to algorytm wnioskowania.

Zauważmy wreszcie, że program dowodzący nie przeszukuje przestrzeni wszystkich możliwych, poprawnie sformułowanych wyrażeń. Z samego algorytmu wynika, że każde nowo utworzone wyrażenie jest dedukowane z aksjomatów. Wyrażenia, które nie spełniają ograniczenia dedukowalności, po prostu nie są generowane. A więc program dowodzący jest także programem spełniającym ograniczenia.

Każde z tych trzech podejść uwypukla inne aspekty procesu rozwiązywania problemów. Przeszukiwanie przestrzeni rozwiązań oraz spełnianie ograniczeń koncentrują się na znajdowaniu rozwiązania danego problemu, natomiast wnioskowanie podkreśla logiczną poprawność połączeń między początkowym stanem problemu a rozwiązaniem. Poszukiwanie rozwiązań to w rzeczywistości odkrywanie, a wnioskowanie to dowodzenie.

Algorytm poszukiwania rozwiązań bada ciągle sytuacje z różnych punktów widzenia, przechodząc od jednego stanu do innego. Asercje, które są prawdziwe w jednej sytuacji (stanie) przestrzeni rozwiązań, nie muszą być prawdziwe w innych sytuacjach. Rozważmy na przykład algorytm wyznaczający kolejne ruchy szachowe. W każdej chwili algorytm koncentruje się na określonej pozycji, w której zachodzą konkretne relacje między figurami szachowymi. Gdy algorytm rozważa nowy ruch, sytuacja ulega zmianie. Relacje, które zachodziły w poprzedniej pozycji, mogą już nie zachodzić; mogły natomiast pojawić się nowe relacje. Nie można już na przykład opierać się na informacji (zgodnej z poprzednim stanem naszej wiedzy), że królowa jest w dalszym ciągu zagrożona lub że wieża nie jest atakowana. Informacja jest zależna od sytuacji i musi być ciągle uaktualniana.

Algorytm wnioskowania gromadzi informacje. Gdy program dowodzenia twierdzeń (pracujący według strategii „do przodu") wykonuje kolejny krok dedukując nowe twierdzenie, wszystkie dowiedzione uprzednio twierdzenia zachowują swą ważność. Wkrótce przekonamy się jednak, że nie wszystkie systemy wnioskujące posiadają właściwość zachowania prawdy i kumulowania twierdzeń.

Artykuł jest tłumaczeniem pracy „Search and Reasoning in Problem Solving" opublikowanej w Artificial Intelligence, Vol. 21, pp. 7–29, 1983.

Herbert A. SIMON jest profesorem informatyki i psychologii w Uniwersytecie Carnegie-Mellon, w którym wykłada od 1949 roku. Od trzydziestu lat zajmuje się badaniami w dziedzinie podejmowania decyzji i rozwiązywania problemów z wykorzystaniem komputerowej symulacji procesów myślowych człowieka.

Tytuł magistra i stopień doktora uzyskał w Uniwersytecie Chicagowskim odpowiednio w latach 1936 i 1943. Od 1967 roku jest członkiem Narodowej Akademii Nauk. Został wyróżniony nagrodami za prace badawcze przez wiele organizacji i stowarzyszeń: American Psychological Association, Association for Computing Machinery (ACM), American Political Science Association, American Economic Association i Institute of Electrical and Electronics Engineers (IEEE). W 1978 roku otrzymał Nagrodę Nobla w dziedzinie ekonomii, a w 1986 roku – Narodowy Medal Nauki. Jest przewodniczącym Zarządu Rady Badań Nauk Społecznych (Social Science Research Council) oraz Sekcji Nauk Behawiorystycznych Narodowej Rady Badawczej (National Research Council). Był również członkiem Komitetu Doradczego ds. Nauki Prezydenta USA.

Prof. H.A. Simon opublikował przeszło 600 artykułów oraz 20 książek i monografii m.in. „Administrative Behavior", „Human Problem Solving" (razem z A. Newellem), „The New Science of Management Decision" (red), „The Science of Artificial", „Models of Thought", „Models of Bounded Rationality", „Reason in Human Affairs", „Protocol Analysis" (razem z K.A. Ericssonem) oraz „Scientific Discovery: Computational Exploration of the Creative Process" (razem z P. Langleyem, G. Bradshawem i J. Zytkowem).

Algorytm spełniający ograniczenia wykonuje duże kroki w procesie rozwiązywania. Nie tworzy on żmudnie kolejnych obiektów na drodze do rozwiązania. Zamiast tego, rozpoczynając pracę od pełnej przestrzeni obiektów, eliminuje on w kolejnych krokach całe klasy obiektów, biorąc pod uwagę zadane ograniczenia. W wyniku następuje zawężenie pierwotnego zbioru do obiektów spełniających wszystkie ograniczenia.

Celem tego artykułu jest przedstawienie zależności między procesami poszukiwania rozwiązań i wnioskowania w rozwiązywaniu problemów, a zwłaszcza przedyskutowanie pewnych aspektów stosowalności i wydajności obu tych podejść w rozwiązywaniu problemów. Bez ambicji pełnego omówienia tematu, w artykule skoncentrowano się na wybranych zagadnieniach, które mają zasadnicze znaczenie w rozwiązywaniu problemów. W szczególności nie omówiono podejścia polegającego na spełnianiu ograniczeń.

Artykuł jest podzielony na trzy części. Pierwsza część jest poświęcona systemom wnioskowania. Druga część omawia systemy przeszukiwania przestrzeni rozwiązań i porównuje przydatność obu tych systemów do rozwiązywania rzeczywistych problemów. Trzecia część zajmuje się zagadnieniami tworzenia reprezentacji praktycznych problemów i ciągłego dopasowywania reprezentacji do zmieniającej się rzeczywistości.

ROZWIĄZYWANIE PROBLEMÓW JAKO WNIOSKOWANIE

Traktowanie rozwiązywania problemów jako procesu wnioskowania znalazło uznanie w oczach wielu badaczy sztucznej inteligencji (zwróćmy na przykład uwagę na rolę, jaką odgrywa rachunek predykatów w podstawowym podręczniku Nilssona [4]). Niektórzy spośród badaczy pod pojęciem wnioskowania rozumieją zastosowanie formalnego systemu logiki, standardowego rachunku predykatów (w dalszej części artykułu przez „standardowy rachunek predykatów" będę rozumieć rachunek predykatów pierwszego lub wyższego rzędu) lub innego rachunku, umożliwiającego opis i rozwiązanie problemu.

Wnioskowanie jako logika

Przełomowymi pracami, dowodzącymi, że proces wnioskowania jest równoważny wykorzystaniu logiki formalnej, były artykuły: McCarthy'ego [2] oraz McCarthy'ego i Hayesa [3]. Ich celem było przede wszystkim rozszerzenie standardowej logiki o dodatkowe mechanizmy wnioskujące, które pozwoliłyby uwzględnić modalne relacje możliwości i przyczynowości, pojawiające się we wnioskowaniu o akcjach.

Dziedzina dowodzenia twierdzeń za pomocą rezolucji zawsze stawiała znak równości między rozwiązywaniem problemów a logiką dedukcyjną, lub mówiąc inaczej – zawsze uważała systemy formalnego dowodzenia twierdzeń za podstawowe narzędzia rozwiązywania problemów. Badania prowadzone w tej dziedzinie ograniczały się zwykle do standardowego rachunku predykatów, przy czym szybko okazało się, że w celu przyspieszenia poszukiwania rozwiązań należy zastąpić zbiory aksjomatów dodatkowymi regułami wnioskowania, pozwalającymi posuwać się w procesie wnioskowania dużymi krokami do przodu.

W ostatnich latach notuje się znaczne zainteresowanie systemami wnioskującymi opartymi na informacjach przechowywanych w dużych bazach danych (często w postaci sieci semantycznych). Problemy, które napotkano w tej dziedzinie, doprowadziły do powstania niestandardowych, na ogół niemonotonicznych logik. Wprowadzenie tego typu logik zostało podyktowane koniecznością możliwie szybkiego kończenia procesów wnioskowania (tzn. przed przejrzeniem wszystkich możliwych ścieżek), koniecznością definiowania wartości standardowych w celu uzupełnienia brakujących informacji i wreszcie koniecznością rozwiązywania sprzeczności pojawiających się w trakcie stosowania niestandardowych procedur wnioskujących.

Wspólną cechą wszystkich tych prac badawczych jest przekonanie, że procesy wnioskowania są w jakiś sposób równoważne stosowaniu logiki dedukcyjnej. Wiarygodność tego punktu widzenia wydaje się potwierdzać potoczny język angielski przez bliskość znaczeniową słów *reasonable* (rozsądny, sensowny) i *logical* (logiczny). Czasem wymaga się nawet, zwłaszcza w dowodzeniu twierdzeń, aby stosowany system logiki spełniał testy zupełności. Aczkolwiek na ogół przedkłada się standardowy rachunek predykatów nad inne formalizmy, to jednak praktyczne potrzeby wymusiły tutaj pewne odstępstwa. Wymieniłem już trzy rodzaje odstępstw (lub rozszerzeń): dodanie operatorów modalnych, rozszerzenie reguł wnioskowania w celu uproszczenia systemów aksjomatów oraz dopuszczenie niemonotoniczności.

Wnioskowanie w matematyce

Wnioskowanie, z którym mamy do czynienia w matematyce, a zwłaszcza w matematyce stosowanej, nie ogranicza się wyłącznie do jawnych aksjomatów i reguł wnioskowania rachunku predykatów, lecz posługuje się znacznie skuteczniejszymi procedurami wnioskowania. Na przykład, równanie liniowe można przekształcić w taki sposób, aby jedna ze zmiennych występujących w równaniu była wyrażona jako funkcja pozostałych zmiennych. Nie musimy przy tym uzasadniać każdego kroku przekształcenia odwołując się do aksjomatów arytmetyki i algebry (aczkolwiek przypuszczalnie moglibyśmy uzasadnić w ten sposób każdy krok). Aby wykonywać takie przekształcenia, definiuje się po prostu zbiory stosownych operatorów („makrooperatorów") i posługuje się nimi rozwiązując równania.

Można oczywiście definiować operatory tego rodzaju formalnie, stosując aksjomatyzację wyższego poziomu. Można również wprowadzać je do systemu wnioskowania za pomocą takiego mechanizmu, jak „uzupełnienie semantyczne" Weyhraucha [6], które pozwala konstruować dowolnie złożone programy służące do modelowania składniowych struktur systemu formalnego na drodze semantycznej. Jednakże w takich hybrydowych rozwiązaniach składniki semantyczne mają raczej właściwości systemów przeszukiwania przestrzeni rozwiązań, a nie systemów wnioskowania i gdy zaczynają one dominować w pracy systemu logicznego, pojawia się wątpliwość, do czego właściwie służą istniejące formalne konstrukcje składniowe. Ta sama uwaga dotyczy propozycji wprowadzania do systemów wnioskowania „uzupełnień proceduralnych"[4]. Uzupełnienia proceduralne są to dowolnie złożone operatory, które dodaje się do systemów formalnych.

Warto w tym miejscu przeanalizować pewien przykład rozwiązywania problemów z wykorzystaniem wnioskowania matematycznego, w którym nie stosuje się bezpośrednio logiki formalnej leżącej w rzeczywistości u podstaw rozwiązywanego problemu (dokładniejsza analiza tego przykładu jest zawarta w [5]). Ten szczególny przykład można widzieć jako szukanie rozwiązania bądź też jako wnioskowanie (zwłaszcza jako wnioskowanie). Proces szukania rozpoczyna się od znanych początkowo przesłanek i zmierza do żądanego wniosku. Proces wnioskowania polega na gromadzeniu wiedzy, aż wartości żądanych wielkości zostaną wyznaczone. W wypadku systemów rozwiązujących problemy przez gromadzenie wiedzy przyjmuje się milcząco założenie, że zdanie raz uznane za prawdziwe pozostaje prawdziwe do końca procesu wnioskowania.

Rozważmy następujący problem:
Częściowo napełniona zlewka zawiera jeden litr 90-procentowego roztworu alkoholu. Ile wody należy dolać, aby otrzymać 80-procentowy roztwór alkoholu?

Ze sformułowania problemu wiemy, że zachodzą pewne zależności (np. początkowa objętość alkoholu równa się 0,9 początkowej objętości cieczy), oraz że dane są pewne wielkości (np. początkowa objętość cieczy wynosi jeden litr). Inne wielkości są początkowo nie znane (np. objętość wody, którą należy dolać).

Informacje podane jawnie w sformułowaniu problemu nie wystarczą jednak do jego rozwiązania. Osoba rozwiązująca musi posłużyć się nie tylko równaniami ze sformułowania problemu, lecz także dodatkową wiedzą. Zakłada się, że osoba rozwiązująca „wie" (niezależnie od tego, czy zostało to potwierdzone doświadczeniem, czy też nie), że objętości cieczy można do siebie dodawać i że na przykład końcowa objętość cieczy jest równa objętości jednego litra plus objętość dolanej wody.

W wypadku tego nadzwyczaj prostego problemu (który zresztą nie jest wcale taki prosty dla większości studentów algebry) można napisać co najmniej dziesięć równań:
(1) $A_1 + dA = A_2$
(2) $W_1 + dW = W_2$

(3) $T_1 + dT = T_2$
(4) $A_1 + W_1 = T_1$
(5) $dA + dW = dT$
(6) $A_2 + W_2 = T_2$

gdzie A, W i T oznaczają odpowiednio alkohol, wodę i roztwór, indeksy 1 i 2 – objętości przed i po dolaniu wody, a d – dodawane objętości. Oprócz tego mamy

(7) $A_1 = 0{,}9\ T_1$
(8) $A_2 = 0{,}8\ T_2$

Ponieważ do roztworu nie dolewamy alkoholu, to

(9) $dA = 0$

Przyjmujemy również, że

(10) $T_1 = 1$

Prosty system produkcji mógłby rozwiązać każde z tych równań ze względu na pozostałą niewiadomą, jeśli tylko w równaniu byłyby znane wartości wszystkich zmiennych z wyjątkiem tej jednej. System taki mógłby szybko obliczyć wartość dW. Na przykład, ponieważ T_1 jest znane z (10), system może rozwiązać (7) ze względu na A_1 i podstawić tę wartość oraz $dA = 0$ z (9) do (1), a następnie rozwiązać to równanie ze względu na A_2. Podstawiając otrzymaną wartość do (8) otrzymujemy wartość T_2, która podstawiona razem z wartością T_1 znaną z (10) do (3) da dT. Ponieważ obie wartości dA i dT są teraz znane, równanie (5) możemy rozwiązać ze względu na dW.

Przedstawiona wyżej ścieżka rozwiązania nie jest jedyna; w rzeczywistości można wyznaczyć 21 różnych sekwencji, z których każda prowadzi do rozwiązania po wykonaniu od pięciu do siedmiu kroków. Różne sekwencje można wygenerować modyfikując nieznacznie system produkcji. Zauważmy, że w wypadku powyższego problemu nie zagraża nam niebezpieczeństwo wykładniczej eksplozji mogącej wystąpić w trakcie niekontrolowanego procesu przeszukiwania przestrzeni rozwiązań. Początkowo mamy 10 równań, z których nie wszystkie są od siebie niezależne. Za każdym razem, gdy znajdujemy wartość jakiejś zmiennej, zmienna ta może zostać usunięta z listy zmiennych do obliczenia, a jej wartość można podstawić do wszystkich równań. Ciąg takich czynności trzeba wykonać tylko raz dla każdej zmiennej, rozwiązując jedynie pewien podzbiór równań. W miarę jak są wyznaczane kolejne niewiadome, liczba czynności, które musi wykonać system, ulega szybkiemu zmniejszeniu; w końcu zostanie wyznaczone pożądane rozwiązanie. Tajemnica zbieżności tego procesu leży w tym, że system nie tworzy nowych zmiennych poza zmiennymi ustalonymi przez początkową reprezentację.

Jest to oczywiście banalny przykład[1], chociaż przejrzenie podręczników fizyki (kinematyki i termodynamiki) pozwala stwierdzić, że nie jest to problem nietypowy dla nauk ścisłych. Specjaliści rozwiązują na ogół problemy tego rodzaju według strategii „do przodu" stosując metody, które wymagają niewiele lub nie wymagają w ogóle wstępnego planowania i które są bardzo podobne do wspomnianego wyżej systemu produkcji.

W powyższym przykładzie z algebry jawne sformułowanie problemu zawiera tylko trzy z dziesięciu niezbędnych równań (równania (7), (8) i (10)). Pozostałe równania, przyjmujące na ogół postać praw zachowania, są dostarczane przez osobę rozwiązującą, która tym samym ogranicza przestrzeń problemów, a następnie poszukuje w niej rozwiązania[2]. Tak więc sformułowanie problemu nie niesie pełnej informacji potrzebnej do rozwiązania problemu. Od osoby rozwiązującej wymaga się dodatkowej wiedzy o wielu założeniach niezbędnych do rozwiązania problemu. Wiedzę tę John McCarthy i inni nazywają „zdrowym rozsądkiem". W rzeczywistości w naszym przykładzie (jak również w wielu innych podręcznikowych problemach) oczekuje się, że rozwiązujący uczyni pewne, sprzeczne z doświadczeniem założenie, jako że objętość mieszaniny wody i alkoholu nie jest dokładnie równa sumie objętości wody i objętości alkoholu. Podobnie w podręcznikowych przykładach o spadających ciałach na ogół zakłada się, że spadanie odbywa się w próżni.

Należy z całą mocą podkreślić, że dodatkowe założenia, które musi zrobić osoba rozwiązująca dany problem, biorą się z doświadczenia i wiedzy tej osoby o świecie. Dlatego też bezsensowne byłyby próby uwzględnienia tych założeń w logice stosowanej do rozwiązywania problemów[3]. Prowadziłoby to oczywiście do błędów w rozwiązywaniu problemów, które opierają się na innych założeniach empirycznych. Nic nie zastąpi osoby rozwiązującej, która wie, jakie założenia można zrobić w konkretnym kontekście rozwiązywanego problemu. W dalszych częściach artykułu omówimy dokładniej, w jaki sposób można wiedzę tego rodzaju uwzględniać w procesie rozwiązywania problemów.

Wnioski z przykładu

Z omówionego wyżej przykładu można wyciągnąć kilka wniosków. Po pierwsze, przez zamianę reprezentacji matematycznej w tego rodzaju problemach na bardziej formalną reprezentację opartą na rachunku predykatów nie osiągnie się żadnych korzyści. W rzeczywistości traci się nawet więcej niż można by zyskać: traci się mianowicie siłę reguł wnioskowania zwyczajnej matematycznej argumentacji, która pozwala zredukować przestrzeń rozwiązań do banalnych rozmiarów i znaleźć rozwiązanie za pomocą bardzo prostego mechanizmu przeszukiwania. Oczywiście stosując uzupełnienia semantyczne i proceduralne, można – jak już wspomniano – wprowadzić reguły wnioskowania matematycznego do systemu formalnego niejako przez „drzwi kuchenne". Wykazałem już jednak, że wejście przez „drzwi kuchenne" jest równoważne zastąpieniu formalnego wnioskowania przez poszukiwanie rozwiązań i modelowanie. Należy więc wystrzegać się stawiania znaku równości między „wnioskowaniem" i stosowaniem rachunku predykatów.

Drugi ważny wniosek płynący z przedstawionego przykładu mówi, że problemy formułowane w ten sposób wymagają na ogół przyjęcia pewnych istotnych założeń implicite, i że dotyczy to nie tylko założeń logicznych (mówiących o tym, jakie dopuszcza się reguły wnioskowania), lecz także przesłanek dotyczących faktów. System wnioskowania posługujący się zdrowym rozsądkiem musi zapewniać możliwość uwzględniania odpowiednich założeń o faktach.

Niedeterministyczny charakter logiki

Stwierdzono już, że procesy przeszukiwania zajmują się odkrywaniem, a procesy wnioskowania – dowodzeniem. Reguły logiki mają charakter przyzwalający: mówią o tym, jakie wnioski mogą być wyciągnięte bezpośrednio ze zbioru przesłanek, a nie – jakie wnioski muszą być wyciągnięte lub też w jakiej kolejności. Logika, jako taka, może być uważana za niedeterministyczny algorytm, który generuje (na przykład przeszukując przestrzeń rozwiązań wszerz) wszystkie konsekwencje wynikające z dawnego zbioru przesłanek.

Jeśli jesteśmy zmuszeni skoncentrować się na określonym podzbiorze wniosków (tzn. jeśli wnioski, którymi jesteśmy zainteresowani, stanowią zaledwie kilka elementów dużej przstrzeni logicznych konsekwencji), to logikę musimy uzupełnić o pewną dodatkową strukturę kontrolną. Na logikę należy wówczas „nałożyć" pewną strategię przeszukiwania. Tak więc, na przykład system w rodzaju Prologu składa się zarówno z logiki, jak i z algorytmu przeszukującego, który działa według strategii do tyłu. W tym sensie procesy wnioskowania stanowią podzbiór procesów przeszukiwania – podzbiór, który jako jedyne operatory stosuje reguły wnioskowania.

Niedeterministyczny charakter logiki nie był specjalnie widoczny w omawianym przykładzie ze względu na mały rozmiar przestrzeni logicznych konsekwencji i ze względu na stosunkowo duże „kroki" operatorów algebraicznych w procesie przekształcania wyrażeń.

[1] Problem ten można jeszcze bardziej zbanalizować, „ukrywając" niektóre z jawnych założeń. Na przykład, jeśli podstawimy $A_1 = A_2 = A$ (objętość alkoholu nie ulega zmianie) i jeśli podstawimy równanie (3) do postaci $T_1 + dW = T_2$, to otrzymamy cztery równania (3), (7), (8) i (10), które rozwiążemy kolejno ze względu na T_1, A_2, T_2 i dW.

[2] W uproszczonej wersji opisanej w poprzednim przypisie, jednym założeniem wykorzystującym prawo zachowania jest równanie (3), a drugim – fakt zrównania A_1 z A_2 na podstawie równań (1) i (9).

[3] Oczywiście powyższa argumentacja jest skierowana przeciwko umieszczaniu takich założeń w aksjomatach logiki, a nie przeciwko wyrażaniu ich w rachunku predykatów lub jakimś innym języku logiki.

dokończenie na s. 7

JANUSZ RYBNIK
JERZY SOLAK
Instytut Informacji Naukowej, Technicznej i Ekonomicznej
Warszawa

POLSKIE TOWARZYSTWO INFORMATYCZNE

Pierwsza wersja artykułu została przedstawiona podczas IV Szkoły Mikrokomputerowej PTI „Mikrokomputery 16- i 32-bitowe"; Łódź, 15-17 grudnia 1987 r.

Stacje robocze Sun-3 (1)

Architektura

W artykule przedstawiono rodzinę komputerów Sun-3 (firmy Sun Microsystems), będącą najbardziej typowym i cieszącym się dużą popularnością na rynku reprezentantem nowej kategorii systemów komputerowych, nazywanych stacjami roboczymi lub stacjami graficznymi. Są one wynikiem ewolucji mikrokomputerów w kierunku systemów o znacznie zwiększonej mocy obliczeniowej, wyposażonych w oprogramowanie systemowe o najwyższym standardzie. Pierwsza część artykułu zawiera omówienie typowych cech stacji roboczych, prezentację poszczególnych modeli rodziny Sun-3 i opis najbardziej charakterystycznych cech ich architektury. Część druga jest poświęcona oprogramowaniu komputerów Sun-3, ze szczególnym uwzględnieniem najbardziej rozpowszechnionego w stacjach roboczych systemu operacyjnego Unix.

PRZEGLĄD STACJI ROBOCZYCH

Komputery Sun pojawiły się na rynku na przełomie lat osiemdziesiątych. Mimo pewnych podobieństw do minikomputerów, mikrokomputerów i terminali graficznych, trudno je zaklasyfikować do jednej z tych rodzin.

Do minikomputerów zbliża je znaczna moc obliczeniowa, której miernikami są:
- pojemność pamięci operacyjnej rzędu kilku lub nawet kilkunastu megabajtów,
- zastosowanie procesorów 32-bitowych (16-bitowe należą do rzadkości), wspomaganych często przez koprocesory arytmetyczne i graficzne, pracujących przy wysokich częstotliwościach zegara.
- bogata architektura wewnętrzna, zapewniająca autonomię i równoległe wykonywanie różnych operacji,
- wielodostęp i wielozadaniowość systemu operacyjnego.

Podobieństwa stacji roboczych do sprzętu mikrokomputerowego przejawiają się przede wszystkim w:
- ogólnej zasadzie konstrukcji, wynikającej z zastosowania mikroprocesora w roli procesora centralnego,
- wyraźnym uprzywilejowaniu jednego użytkownika, który najczęściej kontaktuje się z komputerem w trybie konwersacyjnym.

Cechy charakterystyczne, upodobniające stacje robocze do terminali graficznych to:
- ukierunkowanie na graficzną formę wprowadzania i wyprowadzania danych, co znajduje wyraz w zastosowaniu wysokiej jakości monitorów graficznych, digitizerów, myszy i innych urządzeń ułatwiających graficzną prezentację danych,
- wyposażenie w sprzęt i oprogramowanie, umożliwiające dołączanie do sieci komputerowej.

W przeciwieństwie do terminali graficznych, zawsze podporządkowanych komputerowi macierzystemu, stacje robocze są komputerami niezależnymi.

Stacje robocze często traktuje się jako nowy, doskonalszy typ mikrokomputerów. Ma to uzasadnienie zarówno w technologii produkcji, jak i w ogólnym kierunku ewolucji systemów mikrokomputerowych. Trzeba jednak pamiętać, że stacje robocze znacznie przewyższają powszechnie używane mikrokomputery klasy IBM PC/XT/AT mocą obliczeniową, bogatszą architekturą wewnętrzną, możliwością łączenia w sieć i doskonalszymi urządzeniami do przedstawiania informacji graficznej. Toteż w obecnej fazie rozwoju sprzętu komputerowego celowe wydaje się traktowanie ich jako odrębnej kategorii urządzeń.

Stacje robocze ukształtowały się pod wpływem dwóch tendencji rozwojowych informatyki. Pierwsza z nich to postęp technologii, gwarantujący dostępność, po umiarkowanych cenach, coraz doskonalszych elementów i urządzeń elektronicznych. Pod tym względem decydujące znaczenie dla upowszechnienia stacji roboczych miało pojawienie się na rynku rodziny mikroprocesorów MC680xx firmy Motorola oraz rozwój produkcji dysków stałych Winchester o dużych pojemnościach. Drugim czynnikiem wywierającym wpływ na kształtowanie się stacji roboczych był wzrost zapotrzebowania na systemy komputerowo wspomaganego projektowania (CAD, CAE). Wymaganiom stawianym przez te systemy, zwłaszcza konieczność obsługi bogatego sprzętu graficznego, trudno było sprostać przy użyciu średniej wielkości komputerów pracujących w trybie wielodostępu. Kiedy stało się możliwe udostępnienie znacznej mocy obliczeniowej pojedynczemu użytkownikowi za umiarkowaną cenę, systemy CAD były w naturalny sposób predestynowane do skorzystania z tej szansy. Toteż początkowo właśnie one stanowiły najważniejszą dziedzinę zastosowań stacji roboczych. Obecnie można zaobserwować szybkie rozszerzanie się oferty oprogramowania użytkowego dla stacji roboczych, zwłaszcza w dziedzinach wymagających utrzymywania bezpośredniego kontaktu między użytkownikiem a systemem komputerowym.

Pionierami w dziedzinie tworzenia koncepcji i produkcji stacji roboczych były firmy Apollo Computer i Sun Microsystems. Szybko dołączyli do nich potentaci przemysłu komputerowego: IBM, DEC, Data General, Control Data, Hewlett Packard i inni. Spośród producentów europejskich na rynek stacji roboczych najwcześniej weszły firmy Norsk Data i Olivetti. Chociaż każdy z producentów proponuje własne rozwiązania sprzętowe i programowe, to można w tej różnorodności dostrzec wyraźne kształtowanie się pewnych standardów. Dotyczy to przede wszystkim procesora centralnego, którym – w przeważającej liczbie wypadków – jest mikroprocesor MC68020. Sporadycznie trafiają się wyjątki od tej reguły, np. mikroprocesor MC68010 (w stacji 310 SPU firmy Hewlett Packard), MicroVax 11 (w stacjach firmy DEC), Intel 80386 (np. w modelu Logician firmy Daisy Systems) czy Intel 80286 (M28 firmy Olivetti). Wśród używanych systemów operacyjnych dominują różne mutacje Unixa. Standardem magistrali wewnętrznej jest VME, choć można również spotkać Multibus (np. w stacjach firm Cadlink i Masscomp), Q-bus (w stacjach firmy DEC) i dość często oryginalne rozwiązania firmowe. Pośród standardów sieciowych przeważa Ethernet, wspomagany przez różnorodne systemy obsługi plików, protokoły komunikacyjne itp.

W skład typowego wyposażenia stacji roboczej wchodzą:
- monitor kolorowy lub monochromatyczny, najczęściej o przekątnej ekranu 19 cali i o wysokiej rozdzielczości,
- klawiatura alfanumeryczna,
- mysz,
- digitizer,
- własna jednostka dyskowa typu Winchester, choć dopuszcza się także konfiguracje bezdyskowe, korzystające z baz danych innych komputerów dołączonych do wspólnej sieci.

Podstawowy zestaw urządzeń może być powiększony o drukarki laserowe i inne urządzenia poligraficzne, plotery, urządzenia do składowania i archiwowania danych (streamery), dodatkowe jednostki dyskowe, specjalizowane urządzenia graficzne itp. W skład rodziny, oprócz stacji roboczych, wchodzą również komputery przeznaczone do roli zarządcy plików w sieci, jak i komputery mogące spełniać obydwie te role jednocześnie.

Rodzinę Sun-3 poprzedziła seria Sun-2 o podobnej architekturze, oparta na mikroprocesorze MC68010. W chwili pojawienia się komputerów Sun-3 jej znaczenie zmalało.

PRZEGLĄD RODZINY SUN-3

Rodzina stacji graficznych Sun-3 jest ukierunkowana na zastosowania konstruktorskie, w systemach:
- komputerowo wspomaganego projektowania (CAD – ang. *computer aided design*),
- komputerowo wspomaganego wytwarzania (CAM – ang. *computer aided manufacturing*),
- komputerowo wspomaganej inżynierii oprogramowania (CASE – ang. *computer aided software engineering*),
- komputerowo wspomaganego konstruowania (CAE – ang. *computer aided engineering*),
- komputerowo wspomaganego przygotowania publikacji (CAP – ang. *computer aided publishing*),
- grafiki komputerowej.

Komputery tej rodziny charakteryzują się otwartą architekturą, opartą na standardach przemysłowych, co oznacza łatwość przystosowywania do współpracy z innymi komputerami, działającymi w różnych środowiskach. Służy temu również bogaty zestaw oprogramowania, obsługującego pracę w sieci i zapewniającego współpracę z innymi systemami.

Centralnym elementem konstrukcyjnym komputerów rodziny Sun-3 jest 32-bitowy mikroprocesor MC68020. Wszystkie komputery dysponują pamięcią wirtualną o przestrzeni adresowej do 256 MB na każdy proces. Głównym urządzeniem wyjściowym jest monitor rastrowy o przekątnej ekranu 19 cali i rozdzielczości 1152×900 punktów (zagęszczenie wynosi 81 punktów na cal). Zależnie od modelu, może to być monitor monochromatyczny, czarno-biały z regulowanym poziomem jasności punktów lub kolorowy. Dla modelu 3/260HM przewidziano monitor monochromatyczny o zwiększonej rozdzielczości, wynoszącej 1600×1280 punktów (zagęszczenie 115 punktów na cal). Monitor ma bezpośrednie szybkie połączenie z buforem ekranu w pamięci operacyjnej. Podstawowymi urządzeniami wejściowymi są klawiatura i mysz.

Wszystkie modele rodziny Sun-3, z wyjątkiem modelu 3/50, są wyposażone w 32-bitową magistralę VME. Wszystkie mają też komutatory, umożliwiające łączenie ich w sieć Ethernet.

Systemem operacyjnym, używanym na stacjach Sun-3 jest wzbogacona wersja Unixa 4.2 BSD, opracowanego na Uniwersytecie Kalifornijskim w Berkeley. System ten zapewnia komunikację między procesami i możliwość łączenia komputerów w sieć. W skład oprogramowania podstawowego wchodzą także kompilatory języka C, Pascala, Fortranu 77 i języka asemblera. Wraz z Unixem producent dostarcza bogaty zestaw narzędzi programowych do przetwarzania tekstów, przygotowania dokumentów i opracowania programów, jak również pakiet obsługi plików w sieci (NFS – ang. *network file system*), sieciowy protokół dyskowy i sprzęg użytkowy, obejmujący system okien i biblioteki standardowych procedur graficznych.

Komputery serii Sun-3 mogą być przystosowane do pełnienia różnych funkcji w sieci, bądź też mogą pracować jako jednostki samodzielne. Należy wyróżnić trzy typowe konfiguracje:
- samodzielna stacja robocza, wyposażona we własną pamięć dyskową, pracująca niezależnie lub jako część sieci,
- stacja robocza bez dysku, korzystająca z pamięci dyskowej innych komputerów, współpracujących z nią w sieci,
- komputer serwisowy (ang. *server*, usługodawca), obsługujący wiele terminali i stacji roboczych dołączonych do wspólnej sieci; odpowiednio skonfigurowany usługodawca może obsługiwać od 1 do 20 bezdyskowych klientów.

Niektóre modele były projektowane do pełnienia z góry określonych funkcji, np. modele 3/50 i 3/52 jako stacje robocze, samodzielne albo działające w sieci, a modele 3/180S i 3/280S jako komputery serwisowe. Rodzina Sun-3 wykazuje jednak dużą elastyczność, pozwalającą użytkownikowi kształtować różne konfiguracje, zależnie od indywidualnych potrzeb. Na przykład, model 3/160 może pracować jako samodzielna stacja robocza i jednocześnie obsługiwać pewną liczbę terminali oraz udostępniać swe zasoby innym stacjom, działającym w tej samej sieci.

Na rodzinę Sun-3 składają się trzy serie komputerów.

Seria Sun-3/50

Składa się ona ze stacji Sun-3/50 i Sun-3/52. Standardowo mają one 4 MB pamięci operacyjnej, wbudowany komutator sieci Ethernet i adapter magistrali SCSI (ang. *small computer system interface*). Stacje należące do tej serii pracują przy częstotliwości zegara 15 MHz, z czasem cyklu pamięci 267 ns, co zapewnia sprawność rzędu 1,5 MIPS (1,5 miliona instrukcji na sekundę). Opcjonalnie mogą zostać wyposażone w koprocesor zmiennoprzecinkowy MC68881. Są to typowe biurkowe (ang. *desktop*) stacje robocze, przeznaczone do obsługi jednego użytkownika, z którym komunikują się za pośrednictwem monitora monochromatycznego.

Modele 3/50, 3/52, jak również model 3/75 należący do serii Sun–3/100, mogą współpracować przez łącze SCSI z jedną lub dwiema stacjami dysków o pojemności 71 MB (po sformatowaniu). Firma Sun dostarcza takie stacje w zestawie z pamięcią kasetową o pojemności 60 MB lub bez niej.

Seria Sun–3/100

W jej skład wchodzą modele Sun–3/75, Sun–3/160 i Sun3/180. Standardowo mają one 2 MB pamięci operacyjnej, rozszerzalnej do 8 MB w stacji Sun–3/75 i do 16 MB w modelach Sun–3/160/180. Częstotliwość zegara wynosi dla tej serii 16,67 MHz, a czas cyklu zegara pamięci – 270 ns, dzięki czemu uzyskuje się sprawność 2 MIPS. Wszystkie modele są standardowo wyposażone w koprocesor zmiennoprzecinkowy MC68881.

Model 3/75 jest komputerem stołowym, z założenia przeznaczonym do pracy w sieci. Opcjonalnie można do niego dołączyć pamięć masową. W porównaniu z modelami 3/50 i 3/52 zawiera on dodatkowe złącze magistrali, które może być użyte do rozszerzenia pamięci operacyjnej lub dołączenia dostarczonych przez użytkownika kart o standardzie VME.

Model 3/160 jest komputerem szafkowym, wolno stojącym, montowanym w 12-szczelinowych ramach. Opcjonalnie może zostać wyposażony w akcelerator operacji zmiennoprzecinkowych i kartę multipleksera, umożliwiającą obsługę do 16 terminali. Jest dostępny w czterech wersjach, przystosowanych do pełnienia różnych funkcji. Wersje 3/160M, 3/160C i 3/160G różnią się rodzajem zastosowanego monitora. Model 3/160M współpracuje z monitorem monochromatycznym, 3/160C – z monitorem kolorowym, a 3/160G – z monitorem czarno-białym o 256 poziomach szarości. Wersja 160C daje paletę ponad 16 milionów kolorów, z których 256 może być wyświetlanych jednocześnie, z uwzględnieniem ośmiu poziomów jaskrawości. Wersja ta może pracować z opcjonalnym procesorem graficznym i buforem graficznym. Oprócz wymienionych trzech wersji, model 3/160 występuje w wersji 3/160S, przewidzianej jako komputer serwisowy dla użytkowników działających w sieci. Do podobnych zadań jest przeznaczony model 3/180S, montowany w szafkach z wydajniejszym chłodzeniem i zasilaniem, co ułatwia dodawanie nowych opcji bez ograniczeń stawianych przez model 160S (np. liczbę terminali obsługiwanych asynchronicznie można w nim zwiększyć do 48).

Modele 3/160 i 3/180 pozwalają na dołączenie przez sterownik SMD (ang. *storage module drive*) do 4 dysków o pojemności 130 lub 380 MB, co w maksymalnej konfiguracji zapewnia około 1,5 GB pamięci. Oprócz tego, można dołączyć do 2 stacji pamięci kasetowej typu streamer o pojemności 60 MB każda.

Składają się na nią modele Sun-3/260 i Sun-3/280, będące odpowiednikami modeli 3/160 i 3/180 serii Sun-3/100, o bogatszej architekturze i zwiększonej mocy obliczeniowej. Procesor centralny pracuje w nich z częstotliwością zegara 25 MHz, a zmiennoprzecinkowy koprocesor MC68881 – należący do standardowego wyposażenia – z częstotliwością 20 MHz. Procesor centralny jest wspomagany przez szybką pamięć buforową o pojemności 64 KB, która pozwala prawie do zera zredukować przestoje procesora. Wysokie częstotliwości zegara i pamięć buforowa gwarantują sprawność 4 MIPS.

Modele serii Sun-3/200 są montowane w 12-szczelinowych szafkach, umożliwiających łatwą rozbudowę konfiguracji przez dodawanie nowych kart. Podstawowa konfiguracja obejmuje pamięć operacyjną o pojemności 8 MB, rozszerzalną w razie potrzeby do 16, 24 lub 32 MB.

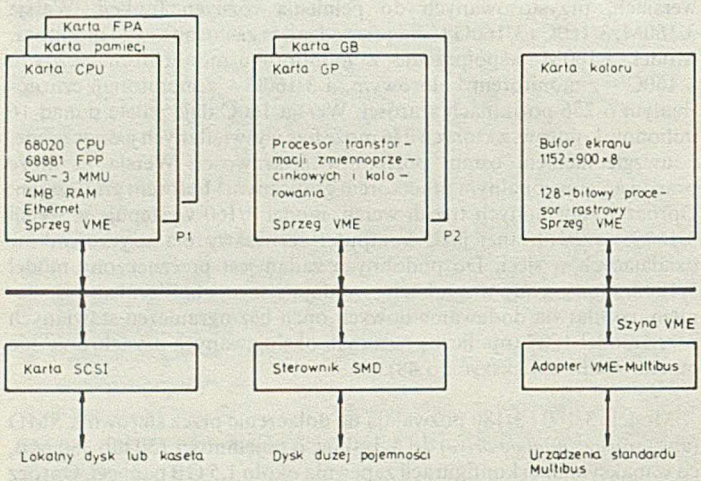
Podobnie jak model 3/160 serii Sun-3/100, model 3/260 występuje w czterech wersjach. Trzy z nich, 3/260 HM, 3/260C i 3/260G, są przeznaczone do pełnienia funkcji stacji roboczych monochromatycznych (HM), kolorowych (C) i czarno-białych ze zmiennym poziomem szarości (G). Wersja 3/260HM zapewnia najwyższą w całej rodzinie Sun-3 rozdzielczość obrazu, wynoszącą 1600 × 1280 punktów. Czwarta wersja modelu 3/260, opatrzona dodatkowym symbolem S, jest przewidziana do pełnienia funkcji komputera serwisowego. Wszystkie wersje tego modelu mogą obsługiwać do 4 stacji dysków o pojemności 280 MB każda. Pod tym względem przewyższa je Sun-3/280S, również zaprojektowany jako komputer serwisowy, przystosowany do współpracy z czterema jednostkami pamięci dyskowej po 585 MB, co w maksymalnej konfiguracji daje imponującą pojemność pamięci masowej rzędu 2,3 GB. Model ten może obsługiwać także do 48 terminali.

Modele serii Sun-3/200 pozwalają na dołączenie dwóch jednostek pamięci kasetowej typu streamer o pojemności 60 MB.

ARCHITEKTURA SPRZĘTU RODZINY SUN-3

Komputery rodziny Sun-3 charakteryzują się otwartą architekturą, której podstawę tworzy magistrala VME. Poszczególne modele są konstruowane przez odpowiedni dobór kart i urządzeń zewnętrznych, wybranych ze standardowego zestawu. Jako przykład zostanie omówiona konfiguracja kolorowej stacji graficznej Sun 3/160C, zawierająca większość kart używanych w ramach serii Sun-3/100. Schemat blokowy tego komputera przedstawiono na rysunku 1. Wyróżnia się w nim bloki:

- karty procesora centralnego (CPU – ang. *central processing unit*),
- karty dodatkowej pamięci,
- akceleratora operacji zmiennoprzecinkowych (FPA – ang. *floating point accelerator*),
- procesora graficznego (GP – ang. *graphics processor*),
- bufora graficznego (GB – ang. *graphics buffer*),
- karty koloru,
- lokalnej pamięci masowej.

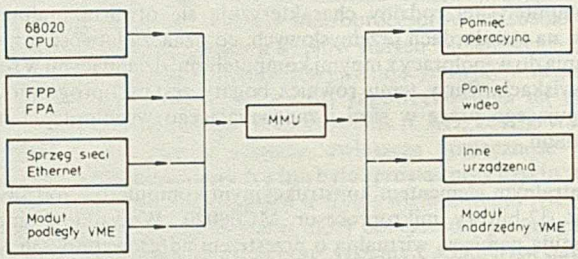


Rys. 1. Architektura systemu Sun-3/160C

Karta CPU zawiera podstawowe elementy stacji, tj. procesor centralny CPU, koprocesor zmiennoprzecinkowy FPP, moduł zarządzania pamięcią MMU, pamięć operacyjną, sprzęg sieciowy i standardowe wejście-wyjście. Karta ta ma swoją własną lokalną magistralę *P*1, zapewniającą szybkie połączenie między poszczególnymi elementami i umożliwiającą dołączanie dodatkowych kart pamięci i akceleratora operacji zmiennoprzecinkowych. Sama karta CPU wystarcza do obsługi kompletnej stacji monochromatycznej.

Inne moduły systemu, takie jak bufor ekranu kolorowego, procesor graficzny oraz sterowniki SCSI i SMD, kontaktują się z CPU i wzajemnie między sobą przez magistralę VME. Moduły mogą komunikować się ze sobą również przez magistrale lokalne. W omawianym systemie, oprócz magistrali *P*1, ten typ połączenia istnieje między procesorem graficznym a buforem graficznym (magistrala lokalna *P*2). Wielomagistralowa architektura stacji Sun-3 umożliwia równoległe przesyłanie danych przez magistrale lokalne i magistralę główną, co znacznie zwiększa efektywność systemu.

Na rysunku 2 przedstawiono architekturę karty CPU. Jej centralnym elementem jest moduł zarządzania pamięcią MMU, tłumaczący wszystkie adresy używane w systemie. Procesor centralny przesyła adres wirtualny do MMU, który przekształca go na adres fizyczny, pozwalający na dostęp do pamięci operacyjnej, pamięci wideo, wejścia-wyjścia lub układów modułu nadrzędnego magistrali. Wszystkie adresy w systemie, przechodzące przez MMU, są tłumaczone i chronione w identyczny sposób. Stosuje się dwupoziomowe tłumaczenie adresu, z użyciem adresu segmentu i strony. Możliwe jest równoległe odwzorowanie adresów ośmiu procesów, z których każdy ma przestrzeń adresów wirtualnych do 256 MB.



Rys. 2. Architektura karty CPU

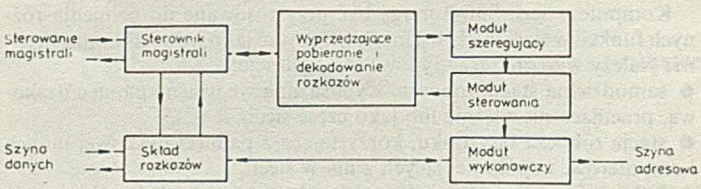
Oprócz urządzeń zaznaczonych na rys. 2, karta CPU zawiera dodatkowe zespoły: EPROM, EEPROM, zegar czasu astronomicznego, sterownik przerwań, szeregowe wejście-wyjście, wejście klawiatury i mysz oraz rejestry sterowania.

ZESPÓŁ PROCESORA CENTRALNEGO

W skład zespołu procesora centralnego, oprócz samego procesora MC68020, wchodzą:

- procesor zmiennoprzecinkowy (FPP),
- akcelerator operacji zmiennoprzecinkowych (FPA),
- IDPROM,
- zespół rejestrów sterowania.

Mikroprocesor MC68020 zawiera szesnaście 32-bitowych rejestrów uniwersalnych, dwa rejestry stosu, dwa liczniki rozkazów, 256-bajtowy skład rozkazów, szybki 32-bitowy moduł wykonawczy, moduł pobierania i dekodowania wyprzedzającego rozkazów oraz moduł sterownika magistrali. Schemat połączeń bloków funkcjonalnych procesora przedstawiono na rysunku 3.



Rys. 3. Schemat blokowy mikroprocesora MC68020

Sterownik magistrali ładuje rozkazy z szyny. danych do modułu dekodującego i do składu rozkazów. Moduł szeregujący i moduł sterowania są odpowiedzialne za całościowe sterowanie procesorem. Zarządzają one wewnętrznymi szynami, rejestrami i modułem wykonawczym. Mikroprocesor MC68020, dzięki potokowej organizacji przetwarzania, umożliwia równoległe wykonywanie do trzech rozkazów.

Zespół procesora obejmuje koprocesor zmiennoprzecinkowy MC68881, wykonujący działania zgodnie z normą IEEE 754. Zawiera on osiem 80-bitowych rejestrów zmiennoprzecinkowych, trzy 32-bitowe rejestry sterowania i sprzęg między koprocesorem a procesorem centralnym, umożliwiający równoległą pracę obydwu procesorów.

Akcelerator operacji zmiennoprzecinkowych (FPA) zwiększa około czterokrotnie szybkość wykonywania działań zmiennoprzecinkowych w porównaniu z koprocesorem zmiennoprzecinkowym. Jest on wyposażony w 32 rejestry zmiennoprzecinkowe i ma bezpośrednie połączenie z magistralą CPU. Operacje CPU i FPA są wykonywane równolegle. FPA wykonuje działania pojedynczej i podwójnej precyzji według normy IEEE 754. Zapewnia sprzętową realizację podstawowych funkcji arytmetycznych i elementarnych funkcji zespolonych. Dalsze zwiększenie efektywności osiąga się dzięki dwustopniowemu potokowi działań.

Wynikowe programy aplikacyjne mogą być przenoszone między komputerami posiadającymi i nie posiadającymi FPA, jednak osiągnięcie maksymalnej efektywności wymaga ich ponownej kompilacji.

Każda stacja Sun-3 jest wyposażona w odczytywalną programowo pamięć IDPROM – 32-bajtową, niemodyfikowalną pamięć PROM bipolarną. Zawiera ona indywidualne dane dotyczące komputera: typ, numer seryjny, adres Ethernetu i datę produkcji.

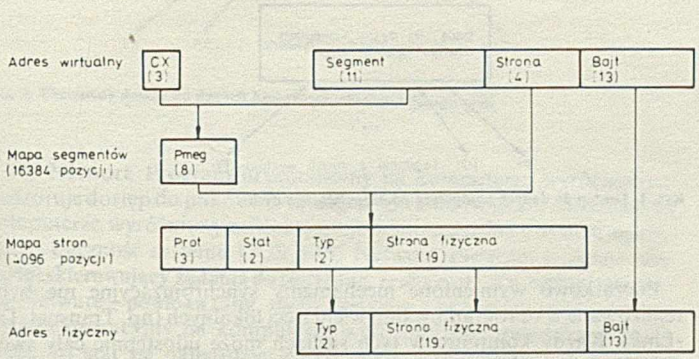
Zespół CPU obejmuje także grupę rejestrów sterowania, używanych do obsługi sytuacji wyjątkowych, ogólnego sterowania i inicjowania systemu. Rejestry te są dostępne bezpośrednio z CPU, bez odwoływania się do MMU.

ARCHITEKTURA PAMIĘCI WIRTUALNEJ

Pamięć wirtualna komputerów Sun-3 umożliwia odwzorowanie do 256 MB pamięci dla pojedynczego procesu. Podstawę działania pamięci stwarza moduł zarządzania pamięcią MMU, opatentowany przez firmę Sun. MMU przechowuje jednocześnie 8 niezależnych kontekstów translacji adresu, między którymi CPU może przełączać się za pomocą jednego rozkazu, bez konieczności przeładowywania dużych tablic.

Urządzenia wejścia-wyjścia, takie jak sprzęg sieci lub dysku, a także koprocesory, używają adresów wirtualnych w komunikacji z pamięcią operacyjną. Ich odwołania do pamięci są tłumaczone i sprawdzane w identyczny sposób jak odwołania procesora centralnego.

Koncepcja pamięci wirtualnej jest w komputerach Sun-3 stosowana także w odniesieniu do urządzeń wejścia-wyjścia. Są one obiektami wirtualnymi, które mogą być odwzorowywane w przestrzeni adresowej użytkownika. Pozwala to na stosowanie w odniesieniu do urządzeń wejścia-wyjścia tych samych mechanizmów ochrony, co dla obszarów pamięci, bez dodatkowego obciążania jądra systemu operacyjnego. Operacje tłumaczenia adresu wirtualnego są wykonywane równolegle z operacjami dostępu do pamięci operacyjnej.



Rys. 4. Schemat translacji adresu wirtualnego na adres fizyczny

MMU składa się z rejestru kontekstu *CX*, mapy segmentów i mapy stron. Adres wirtualny z procesora jest tłumaczony za pomocą mapy segmentów na adres pośredni, który z kolei jest zamieniany przy użyciu mapy stron na adres fizyczny. Strony mają rozmiar 8 KB, a segmenty – 128 KB. Na rysunku 4 przedstawiono schemat translacji adresu wirtualnego na adres fizyczny.

Wybór kontekstu, tzn. procesu, dla którego jest obliczany adres, zależy od zawartości 3-bitowego rejestru kontekstu *CX*. Na podstawie zawartości tego rejestru oraz jedenastu najbardziej znaczących bitów adresu wirtualnego jest identyfikowany jeden z 16 384 segmentów pamięci wirtualnej. Dla każdego segmentu wirtualnego odpowiednia 8-bitowa pozycja mapy segmentów wskazuje na 16-elementową grupę pozycji w mapie stron PMEG (ang. *page map entry group*). Dokładny wybór pozycji w grupie odbywa się na podstawie dalszych 4 bitów adresu wirtualnego. W sumie mapa stron składa się z 4096 pozycji, z których każda wskazuje stronę o wielkości 8 KB. Każda pozycja mapy stron zawiera pole ochrony, pole typu, pole statystyk i numer strony.

Pole ochrony składa się z 3 bitów: bitu ważności, bitu zapisu i bitu programu nadzorczego. Bit ważności wskazuje, że pozycja w mapie stron jest prawidłowa i umożliwia odczytanie zawartości strony. Bit zapisu zezwala na zapis w obrębie strony. Bit programu nadzorczego oznacza, że dostęp do strony jest zastrzeżony dla programu nadzorczego systemu operacyjnego.

Pole typu strony wyróżnia jedną z czterech fizycznych przestrzeni adresowych: pamięć operacyjną, urządzenia wejścia-wyjścia, 16-bitową magistralę VME lub 32-bitową magistralę VME.

Pole statystyk informuje o tym, jakiego rodzaju operacje były wykonywane na zawartości strony. Pierwszy bit pola statystyk jest ustawiany w chwili pojawienia się żądania dostępu do strony, a drugi bit – podczas aktualizowania zawartości strony. Bity statystyk są aktualizowane sprzętowo.

Fizyczny adres strony, łącznie z numerem bajtu pobranym bezpośrednio z adresu wirtualnego, tworzą ostateczny adres fizyczny.

LITERATURA

[1] Machover C.: Engineering Workstations. An Overview. Pp. V19-V37, Proc. ACM SIGGRAPH. Anaheim (CA), July 1987
[2] Rybnik J., Solak J.: Rodzina komputerów Sun-3. Str. 115-137, Materiały IV Szkoły Mikrokomputerowej PTI. Mikrokomputery 16- i 32-bitowe. Łódź, 15-17 grudnia 1987. ZETO, Łódź, 1987
[3] Sun System Overview. Sun Microsystems. February 1986
[4] Sun-3 Architecture. Technical Report, Sun Microsystems, 1985
[5] Sun-3/200 Series. Sun Microsystems, 1986
[6] Systems International, pp. 33-41, April 1987

Procesy przeszukiwania i wnioskowania

dokończenie ze s. 3

Mogliśmy więc uważać, że kolejność wyciągania wniosków jest nieistotna; nie spowodowałoby to bowiem poważnego opóźnienia lub dodatkowych kłopotów, gdyby po drodze wyciągać wnioski nieistotne z punktu widzenia docelowego wniosku. Jeżeli jednak mamy do czynienia z dużą przestrzenią rozwiązań, to musimy skonstruować wydajny algorytm przeszukiwania, kontrolujący kolejność stosowania operatorów.

tłumaczył i opracował
MAREK MACHURA

LITERATURA

[1] Bobrow D.G. (Ed.): Special Issue on Non-Monotonic Logic. Artificial Intelligence, Vol. 13, No. 1–2, 1980
[2] McCarthy J.: Programs with common sense. Proc. of the Teddington Conference on the Mechanization of Thought Processes, H.M. Stationary Office, London, 1960
[3] McCarthy J., Hayes P.: Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligence 4, American Elsevier, New York, 1969
[4] Nilsson N.J.: Principles of Artificial Intelligence. Tioga Press, Palo Alto (CA), 1980
[5] Simon H.A.: The theory of problem solving. Information Processing 71, pp.261–272, North-Holland, Amsterdam, 1972
[6] Weyhrauch R.W.: Prolegomena to a theory of mechanized formal reasoning. Artificial Intelligence, Vol. 13, pp. 130–170, 1980.

JAROSŁAW GŁĘBICKI
Warszawa

System dBase III Plus w sieciach lokalnych

Ważny etap w rozwoju techniki mikrokomputerowej stanowią lokalne sieci komputerowe. Połączenie komputerów w sieci daje możliwość dzielonego dostępu do drogich urządzeń zewnętrznych (dysków, drukarek, ploterów itp.), a przez to pozwala na efektywne wykorzystanie tych urządzeń. Jeszcze cenniejszą właściwością sieci lokalnej jest możliwość jednoczesnego korzystania ze wspólnych danych przez wielu użytkowników pracujących przy różnych komputerach. W zastosowaniach, w których sposób przetwarzania danych i wyniki zależą bezpośrednio od zawartości bazy danych, zaś baza danych może być zmieniana w dowolnej chwili przez kilku użytkowników, wspomniana cecha sieci lokalnych jest szczególnie istotna. Sieć może wówczas stanowić konkurencję dla systemów wielodostępnych. Typowym przykładem takiego zastosowania może być system rezerwacji miejsc z co najmniej dwoma węzłami sieci (mikrokomputerami) do wprowadzania i odczytu informacji.

DOSTĘP WSPÓLNY DO BAZY DANYCH

Dzielenie dostępu do danych pozwala umieszczać je w jednym miejscu, przez co daje się uniknąć redundancji pamiętanej informacji z wszystkimi jej ujemnymi następstwami, a ponadto wprowadzić system ochrony przed niepowołanym dostępem.

W środowisku, w którym kilka równolegle wykonywanych programów (być może jednakowych) w różnych węzłach sieci ma dostęp do wspólnych danych, może dojść do sytuacji prowadzących do niepoprawnego przetwarzania danych, pomimo że programy te są poprawne w środowisku z jednym użytkownikiem.

Rodzaje błędów

Sytuacje błędne można podzielić na następujące trzy rodzaje:

1. Utrata wprowadzonej zmiany do bazy danych.

Załóżmy, że dwa programy $P1$ i $P2$ zwiększają o 20 zawartość pola $X = 40$ określonego rekordu, oraz że wystąpiła sekwencja zdarzeń:

a) $P1$ odczytuje X ($X = 40$)
b) $P2$ odczytuje X ($X = 40$)
c) $P1$ i $P2$ lokalnie zwiększają X o 20
d) $P1$ zapisuje X ($X = 60$)
e) $P2$ zapisuje X ($X = 60$)

W punkcie e) zmiana wprowadzona do bazy danych przez $P1$ została utracona, a pole X ma wartość 60 zamiast 80.

2. Błędny odczyt.

Niech baza danych będzie tak zorganizowana, że dla każdego rekordu zachodzi: $70 < X+Y < 100$, gdzie X i Y są nazwami pól. Program $P1$ zwiększa zawartość pola $X=20$ określonego rekordu o 50 oraz zmniejsza zawartość pola $Y=60$ też o 50. Program $P2$ oblicza $X+Y$ dla tego samego rekordu. Załóżmy, że nastąpiła sekwencja zdarzeń:

a) $P1$ modyfikuje pole X ($X=70$)
b) $P2$ odczytuje X i $P2$ odczytuje Y ($X=70$, $Y=60$)
c) $P1$ modyfikuje pole Y ($Y=10$)

Po wykonaniu tej sekwencji baza danych nadal zawiera poprawne wartości pól X i Y ($X=70, Y=10$). W punkcie b) program $P2$ dokonał błędnego odczytu i mógłby na przykład niepotrzebnie zasygnalizować błąd w bazie danych ($70+60>100$).

3. Ponowny dostęp do danych, które uległy zmianie.

Niech program $P1$ sporządza listę wszystkich rekordów pewnego pliku, a program $P2$ umieszcza w tym pliku nowy rekord.
Załóżmy następującą sekwencję zdarzeń:
a) $P1$ odczytuje liczbę $K=120$ rekordów w pliku i drukuje tę liczbę w nagłówku sporządzanego raportu,
b) $P2$ umieszcza nowy rekord,
c) $P1$ drukuje 121 rekordów.
W punkcie c) program $P1$ uzyskał dostęp do danych, które uległy zmianie, a przez to niepoprawnie sporządził raport.

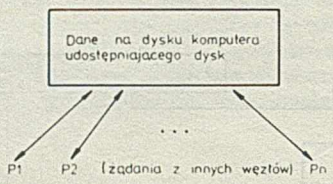
Mechanizmy synchronizacji dostępu

Powyższe wypadki mogą zachodzić równocześnie, a każdy z nich prowadzi zwykle do utraty spójności bazy danych. W oprogramowaniu sieci lokalnej muszą zatem istnieć pewne mechanizmy synchronizacyjne, których właściwe stosowanie chroniłoby przed tego rodzaju sytuacjami. Wyróżnia się dwa takie mechanizmy (wzajemnie uzupełniające się):
- sterowanie trybem dostępu do pliku (ang. *file access control*),
- blokowanie (ang. *locking*).

Sterowanie trybem dostępu do pliku polega na tym, że program otwierając określony plik przekazuje do systemu dwa rodzaje informacji:
– jakie operacje będzie wykonywać na pliku,
– jakie operacje mają być zabronione na czas dostępu tego programu do pliku.
Wówczas program ten uzyska dostęp do pliku, jeżeli:
– operacje, które zamierza wykonywać nie są jeszcze zabronione przez inne programy,
– operacje, które na czas dostępu tego programu mają być zabronione, nie są aktualnie dozwolone dla jakiegoś innego programu. W przeciwnym wypadku program nie będzie mógł otworzyć pliku (zostanie udostępniony kod błędu).

Niech na przykład program $P1$ otwiera plik w celu odczytywania i dopisywania nowych informacji zabraniając modyfikowania pliku. Jeżeli w innym węźle sieci wykonuje się program $P2$, który aktualnie może zapisywać do tego pliku, to program $P1$ nie uzyska dostępu.

Blokowanie jest znanym mechanizmem w wielodostępnych bazach danych. W wypadku sieci lokalnej polega ono na tymczasowym (do chwili odblokowania) niedopuszczeniu innych programów do wykonywania określonych operacji na danym pliku lub rekordzie. Przykładowo, program $P1$ może sporządzać raport i na czas drukowania blokować modyfikacje określonego pliku (lub plików).



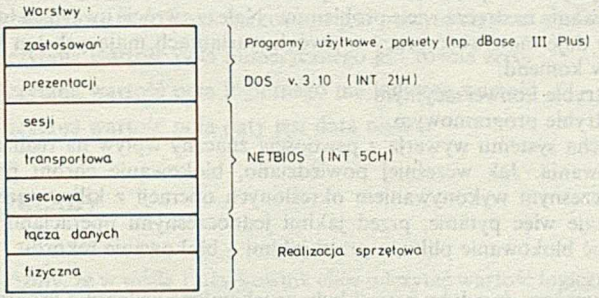
Rys. 1. Dostęp do danych komputera udostępniającego dysk

Początkowo wymienione mechanizmy synchronizacyjne nie były realizowane w oprogramowaniu wielu sieci lokalnych (np. Transnet, D-Link). Każdy komputer w tych sieciach może udostępnić cały swój dysk, innym węzłom. Jest on wówczas tzw. **komputerem udostępniającym dysk** czyli usługodawcą dysku (ang. *disk server*).

Rysunek 1 przedstawia schemat dostępu do danych pamiętanych na dysku takiego komputera. Tego rodzaju rozwiązanie jest oczywiście niewystarczające dla dzielonego dostępu do wspólnych danych. Twórca oprogramowania użytkowego musiałby wtedy tworzyć mechanizmy synchronizacyjne w swoich programach, tym samym uzależniając swoje oprogramowanie od sprzętu konkretnej sieci lokalnej.

Wprowadzenie NETBIOS-a

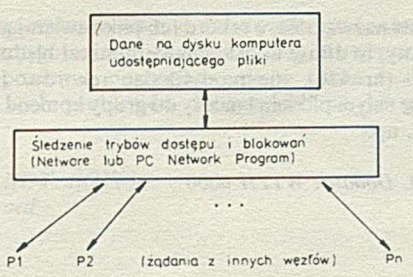
Sytuacja ta uległa zmianie z chwilą pojawienia się standardów w oprogramowaniu sieciowym, NETBIOS i DOS 3.10, których miejsce w warstwowym modelu ISO przedstawiono na rys. 2.



Rys. 2. Warstwowy model ISO dla sieci lokalnej

NETBIOS (ang. *Network Basic Input/Output System*) został umieszczony w pamięci ROMBIOS dla lokalnej sieci komputerowej firmy IBM, stając się automatycznie standardem w odpowiednich warstwach modelu ISO. W systemie operacyjnym DOS 3.10 natomiast znajdują się opisane wcześniej mechanizmy synchronizacyjne. Należy jednak podkreślić, że DOS nie jest systemem wielodostępnym, a realizacja mechanizmów synchronizacyjnych wymaga oprogramowania sieciowego. Powszechnie znane są obecnie dwa, różniące się od siebie, programy zarządzające siecią: PC Network Program firmy IBM i Netware firmy Novell.

Mechanizmy synchronizacyjne wymagają śledzenia dostępu do poszczególnych plików i sposobu blokowania. Dlatego komputer udostępniający dysk został zastąpiony przez tzw. **komputer udostępniający pliki,** czyli usługodawcę plików (ang. file server). Rysunek 3 przedstawia schemat dostępu do danych pamiętanych na dysku takiego komputera. Komputery udostępniające pliki są wyróżnione w sieci. Jest ich na ogół niewiele. Natomiast pozostałe, korzystające z tych plików noszą nazwę stacji roboczych (ang. *workstations)*.



Rys. 3. Chroniony dostęp do danych komputera udostępniającego pliki

PC Network Program uruchomiony na komputerze wyróżnionym nadzoruje dostęp do plików. W oprogramowaniu Netware natomiast na komputerze wyróżnionym funkcjonuje wielozadaniowy system operacyjny. Łączność systemu DOS z PC Network Program realizuje tzw. **moduł skierowujący żądania do sieci** (ang. *redirector*), a w oprogramowaniu Netware – **powłoka** (ang. *shell*). Netware z przyjętym rozwiązaniem (system operacyjny na komputerze wyróżnionym i powłoką w stacji roboczej) stał się najbardziej efektywnym i elastycznym, a przez to najpopularniejszym oprogramowaniem sieci lokalnych (setki tysięcy instalacji w krajach wysoko uprzemysłowionych).

Pojawienie się standardu NETBIOS i DOS 3.10 skłoniło natychmiast firmy produkujące oprogramowanie do tworzenia sieciowych wersji swoich produktów. Typowym tego przykładem jest baza danych dBase III Plus – sieciowa wersja dBase III.

SYSTEM dBase III PLUS

Dostęp do wspólnych danych w systemie dBase III Plus otrzymują użytkownicy uruchamiając ten sam program – dBase Administrator – z komputera udostępniającego pliki (rozwiązanie to praktycznie wyklucza wykorzystanie łączy szeregowych i modemowych z uwagi na czas trwania transmisji całego programu).

Najistotniejszymi elementami dBase III Plus z punktu widzenia zastosowań do baz danych w sieci lokalnej są oczywiście mechanizmy synchronizacyjne.

Sterowanie trybem dostępu do pliku

Istnieją dwa tryby dostępu do każdego pliku dBase:
- **tryb wyłączny** (ang. *exclusive)*,
- **tryb dzielony** (ang. *shared*).

Próba otwarcia danego pliku w trybie wyłącznym oznacza, że użytkownik (lub program) żąda możliwości wykonywania wszystkich dozwolonych przez dBase operacji na tym pliku. Na czas dostępu do pliku (do momentu zamknięcia) wszystkie operacje na nim w innych węzłach mają być zabronione. Dostęp do pliku w trybie wyłącznym zostanie przyznany, jeżeli w żadnym innym węźle plik ten nie jest użytkowany.

Próba otwarcia danego pliku w trybie dzielonym oznacza, że użytkownik (lub program) żąda możliwości wykonywania na pliku operacji określonych (przez dBase) dla tego trybu. Dostęp do pliku w trybie dzielonym zostanie przyznany, jeżeli w żadnym innym węźle plik ten nie jest otwarty w trybie wyłącznym. Określone dla trybu dzielonego operacje zależą od rodzaju pliku. Dla wszystkich plików, z wyjątkiem bazy danych (*dbf*), indeksów (*ndx*) i notatek (*dbt)* otwartych komendą USE, są to operacje odczytu pliku. Tryb dostępu dla tych plików jest ustalony automatycznie na podstawie komendy otwierającej dany plik. Zasada przy tym jest następująca: jeżeli komenda umożliwia utworzenie pliku lub wprowadzanie zmian do istniejącego pliku, to przyjmuje się próbę otwarcia w trybie wyłącznym. W przeciwnym razie (komenda korzysta jedynie z zawartości pliku) przyjmuje się próbę otwarcia w trybie dzielonym.

Rozważmy plik *WYKAZ.PRG* zawierający tekst programu. Komenda *MODIFY COMMAND* powoduje próbę otwarcia tego pliku w trybie wyłącznym, natomiast komenda *DO WYKAZ* powoduje próbę otwarcia pliku w trybie dzielonym. Wynika stąd, że tylko w jednym węźle można modyfikować dany program i w czasie trwania tej modyfikacji program nie może być wykonywany.

Podobnie jest dla innych plików np.:
CREATE LABEL i *MODIFY LABEL* ⟨plik etykiet⟩ otwierają w trybie wyłącznym,
LABEL FORM ⟨plik etykiet⟩ otwiera w trybie dzielonym,
SAVE ⟨plik zmiennych⟩ otwiera w trybie wyłącznym,
RESTORE ⟨plik zmiennych⟩ otwiera w trybie dzielonym,
MODIFY COMMAND ⟨plik procedur⟩ otwiera w trybie wyłącznym,
SET PROCEDURE TO ⟨plik procedur⟩ otwiera w trybie dzielonym,
COPY TO ⟨plik bazy danych⟩ otwiera w trybie wyłącznym,
APPEND FROM ⟨plik bazy danych⟩ otwiera w trybie dzielonym.

Odmienny charakter ma komenda USE. Otwierając plik bazy danych komendą *USE* użytkownik (lub program) decyduje o wyborze trybu dostępu. Otwarte z tym plikiem indeksy lub notatki mają zawsze taki sam tryb dostępu jak dany plik.

Użycie tej komendy w postaci:

USE ⟨nazwa pliku⟩

powoduje przyjęcie domyślnego trybu dostępu. Domyślny tryb dostępu można zmieniać komendą:

SET EXCLUSIVE ON/OFF.

W chwili wywołania dBase przyjęta jest wartość *ON*, czyli domyślny tryb dostępu wyłączny (chyba, że w pliku *CONFIG.DB* jest wstawione *SET EXCLUSIVE OFF*). Domyślny tryb dostępu można pominąć podając jawnie w komendzie USE, że chodzi o tryb wyłączny np.: *USE EXCLUSIVE Kadry.*

Nie można jednak w podobny sposób wymusić dostępu dzielonego.

W trybie wyłącznym można dokonywać wszystkich operacji na pliku. W trybie dzielonym nie można wykonywać następujących komend: *INSERT, INSERT BLANK, MODIFY STRUCTURE, PACK, REINDEX* oraz *ZAP*. Zauważmy, że komendy te zmieniają obraz pliku (w sensie struktury, indeksu, numeracji rekordów itp.) i wymagają trybu wyłącznego. Użycie pozostałych komend w trybie dzielonym może być dodatkowo uzależnione i powiązane z mechanizmem blokowania.

Jeżeli otwarcie dowolnego pliku dla danego trybu dostępu jest niemożliwe, to dBase udostępnia komunikat błędu: *„Plik jest używany przez kogoś innego"* (nr 108).

Blokowanie

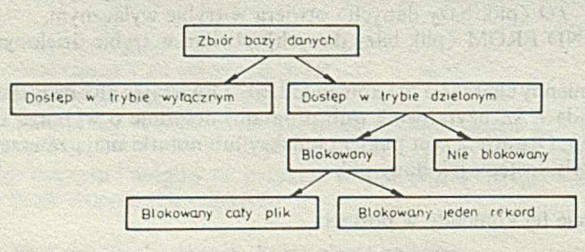
Stosowanie mechanizmów blokowania ma sens tylko dla plików bazy danych otwartych komendą USE w trybie dzielonym. W trybie wyłącznym blokowania są ignorowane. System dBase III Plus daje możliwość zablokowania całego pliku lub jednego rekordu danego pliku – rekordu bieżącego. Do tego celu służą odpowiednio dwie bezparametrowe funkcje logiczne: *FLOCK( )* oraz *RLOCK( )* (lub *LOCK( )*). Funkcje te udostępniają wartość *.F.*, jeżeli pliku lub rekordu nie można zablokować (jest już blokowany przez inny węzeł). Jeżeli natomiast plik lub rekord można zablokować, to udostępniają wartość *.T.* i dodatkowo, jako efekt uboczny, dokonują blokowania. System dBase przechowuje informacje o blokowaniu dla każdego pliku. Dlatego też użytkownik może w danej chwili blokować kilka plików lub kilka rekordów – po jednym w każdym pliku. Na przykład, komendy:

```
.USE Pracownicy
.?RLOCK ( )
.T.
.SELECT 2
.USE Kierownicy
.GO BOTTOM
.? RLOCK( )
.T.
```

powodują zablokowanie dwóch rekordów – pierwszego rekordu w pliku *Pracownicy* i ostatniego rekordu w pliku *Kierownicy*.

System dBase III Plus przestrzega tzw. ziarnistości blokowania, tzn. plik nie może być zablokowany, jeżeli w innym węźle blokowany jest ten plik lub jeden z jego rekordów, oraz odwrotnie – rekord nie może być zablokowany, jeżeli w innym węźle blokowany jest ten sam rekord lub plik, w którym rekord ten jest pamiętany. Odblokowanie rekordu lub pliku następuje komendą *UNLOCK* (dotyczy tylko pliku w bieżącym polu roboczym) lub w chwili dokonywania nowej próby blokowania danego pliku lub jednego z jego rekordów niezależnie od tego, czy próba ta powiedzie się, czy też nie, albo w chwili zamknięcia pliku.

Funkcje *FLOCK( )* i *RLOCK( )* (lub *LOCK( )*) nie są jednak jedynymi mechanizmami blokowania. Następujące komendy operujące całym plikiem automatycznie blokują ten plik (jeśli jest to możliwe) w chwili wywołania i odblokowują po zakończeniu wykonywania: *APPEND, APPEND BLANK, APPEND FROM, AVERAGE, BROWSE, CATALOG, COPY, COPY STRUCTURE, COUNT, DELETE ALL, INDEX, JOIN, RECALL ALL, REPLACE ALL, SORT, SUM, TOTAL, UPDATE*. Ponadto użytkownik może zablokować bieżący rekord oglądany komendą *EDIT* lub *CHANGE* naciskając klawisze *CTRL*-0 i odblokować ten rekord naciskając ponownie *CTRL*-0 lub przechodząc do innego rekordu.



Rys. 4. Schemat mechanizmów synchronizacyjnych dla pliku bazy danych

Na rysunku 4 przedstawiono schemat mechanizmów synchronizacyjnych dla pliku bazy danych. Użycie komendy *DISPLAY STATUS* pozwala odczytać dla każdego pliku bazy danych, w którym miejscu poniższego schematu plik ten się znajduje.

Wprawdzie sterowanie trybem dostępu w dBase III Plus jest proste i przejrzyste (w porównaniu z systemem DOS 3.10), to stosowanie blokowania nastręcza wiele problemów. Należy zwrócić uwagę na to, że dBase daje możliwość pracy w dwóch wariantach mających ten sam zestaw komend:

- w trybie konwersacyjnym
- w trybie programowym.

Ta cecha systemu wywarła z pewnością znaczny wpływ na realizację blokowania. Jak wcześniej powiedziano, blokowanie chroni przed jednoczesnym wykonywaniem określonych operacji z kilku węzłów. Powstaje więc pytanie, przed jakimi jednoczesnymi operacjami ma chronić blokowanie pliku, a przed jakimi – blokowanie rekordu.

Zauważmy, że gdyby w sieci były uruchamiane wyłącznie programy użytkowe, to odpowiedź na to pytanie leżałaby wyłącznie w gestii programistów. Przyjęcie przez nich umowy, przed jakimi operacjami program musi zablokować plik lub rekord, automatycznie determinuje takie operacje. W dBase III Plus rozwiązanie takie nie jest możliwe z uwagi na wariant pracy w trybie konwersacyjnym. Działanie systemu nie może bowiem być uzależnione od umowy między jego użytkownikami. W związku z tym w dBase zostały z góry określone operacje wymagające blokowania pliku lub rekordu. Jeżeli plik lub rekord nie jest blokowany, to dBase podejmuje odpowiednie środki zaradcze. Problem polega na tym, że trudno podać prostą regułę określającą operacje wymagające blokowania i środki zaradcze podejmowane przez dBase.

Blokowania pliku bazy danych wymagają wszystkie operacje modyfikujące plik. Operacje modyfikujące zawartość jednego rekordu wymagają zablokowania rekordu lub pliku. Jeżeli operacja modyfikuje zawartość jednego rekordu, np.:

REPLACE Dodatek WITH 6000

lub

@ *..GET Dodatek*

READ.

gdzie *Dodatek* jest nazwą pola, a rekord lub plik zawierający ten rekord nie jest blokowany, to dBase udostępnia komunikat błędu *„Rekord nie jest blokowany"* (nr 130), nie modyfikując rekordu. Jeżeli jednak komenda operuje całym plikiem i należy do grupy komend automatycznie blokujących, np.:

REPLACE ALL Dodatek WITH 6000

lub

APPEND BLANK

to reakcja systemu jest odmienna. System dBase próbuje wtedy zablokować cały plik (odblokowując przy okazji rekord lub plik, jeżeli jest blokowany). Jeżeli plik można zablokować, to blokuje go, przeprowadza operacje i odblokowuje, w przeciwnym wypadku udostępnia komunikat błędu *„Plik jest używany przez kogoś innego"* (nr 108). Widać zatem, że ta sama komenda (np. *REPLACE*) może powodować różne reakcje systemu.

Kłopoty powstają również podczas odczytywania danych. W ogólnym wypadku odczyt pliku nie wymaga wcześniejszego blokowania. Można również odczytywać plik, gdy w innym węźle jest on blokowany np.:

z_dodatek = Dodatek

gdzie z_dodatek jest zmienną, a *Dodatek* – nazwą pola. Jeżeli jednak komenda działa na cały plik i automatycznie blokuje, np. *COUNT, SUM, TOTAL,* to reakcja dBase jest taka jak przy modyfikacji. Należy wtedy spodziewać się ewentualnego komunikatu błędu albo odblokowania, jeżeli blokowany był plik lub rekord.

Odczytanie zawartości pól zablokowanego rekordu nie jest możliwe. Reakcja systemu jest w tym wypadku dość specyficzna. System dBase udostępnia komunikat błędu *„Rekord jest używany przez kogoś innego"* (nr 109) tylko w momencie, gdy blokowany przez kogoś rekord staje się rekordem bieżącym w wyniku wykonania komendy wrażliwej na kody błędu (np. *GOTO*, ale nie *SKIP*!). Pomimo sygnalizacji błędu rekord ten staje się rekordem bieżącym. W pozostałych wypadkach nie ma żadnej sygnalizacji. Odczytane z takiego rekordu wartości są puste, tzn.:

- odczytana wartość pola znakowego jest pustym ciągiem znaków,
- odczytana wartość pola numerycznego jest równa zero,
- odczytana wartość pola logicznego ma logiczną wartość *.F.*
- odczytana wartość pola daty jest datą nijaką.

PRZYKŁADY

Załóżmy, że w węźle 1 użytkownik chce odczytać wartość logicznego pola *Karany* pierwszego rekordu pliku *Kadry*, a w węźle 2 inny użytkownik wstawia do pola *Imie* napis *„Jan"*. Niech wartość pola *Karany* w pierwszym rekordzie jest *.T.*, a sekwencja zdarzeń jest taka jak przedstawiono na wydruku 1.

	Węzeł 1	Węzeł 2	Uwagi
1)	SET EXCLUSIVE OFF	SET EXCLUSIVE OFF	
2)	USE Kadry	USE Kadry	
3)	GOTO 1	GOTO 1	
4)		? RLOCK()	W węźle 2 udane zablokowanie
5)	? Karany ~.F.	REPLACE Imie WITH 'Jan'	W węźle 2 odczyt .F. zamiast .T. bez żadnej sygnalizacji
6)	USE	USE	

Wydruk 1

Użytkownik lub program musi więc zatroszczyć się o to, aby nie odczytywać pustych wartości, np. przez wcześniejsze zablokowanie rekordu lub wykonanie *GOTO RECNO( )*. Zauważmy również, że zmiana wartości pól danego rekordu jest możliwa przy blokowaniu rekordu lub pliku, jednak blokując przy tym plik umożliwiamy odczyt tych wartości, a blokując rekord, zabraniamy takiego odczytu.

Z powyższego opisu wynika, że stosowanie mechanizmów blokowania wymaga szczególnej ostrożności. Dotyczy to zwłaszcza programów użytkowych, ponieważ programowanie w takim środowisku może prowadzić do wielu problemów. Na przykład, zwykła pętla:

```
GO TOP
DO WHILE .NOT. EOF()
  ...zapisz rekord
  SKIP
ENDDO
```

może prowadzić do odczytu pustej wartości i powinna być zastąpiona, na przykład przez pętlę:

```
ii=1
DO WHILE ii < = RECCOUNT()
  GOTO ii
  ... zapisz rekord
  ii = ii + 1
ENDDO
```

jeżeli plik nie jest indeksowany, z odpowiednią procedurą obsługi błędu. Może też być poprzedzona zablokowaniem pliku lub też zawierać wewnątrz blokowanie kolejnych rekordów.

```
...
DO CASE
...
  CASE ERROR()=108 .OR. ERROR()=109
    * 108 - Plik używany przez kogoś innego
    * 109   Rekord używany przez kogoś innego
    kom=IIF(ERROR()=108 , 'Plik' , 'Rekord')
    kom=kom+'jest niedostępny. SPACJA przerywa czekanie...'
    SET COLOR TO N/W
    @ 24,0 SAY SPACE(14)+kom+SPACE(14)&& Pisz komunikat błędu
    SET COLOR TO
    ?? CHR(7)                          && Sygnały dźwiękowe
    ?? CHR(7)
    ?? CHR(7)
    @ 24,0 SAY SPACE(80)               && Komunikat znika
    IF INKEY()<>32                     && Czy wciśnięto spację
      RETRY                            && Jeśli nie, to ponów
      ELSE                             && operację, a jeśli tak
      RETURN TO MASTER                 && to powróć do menu gł.
    END IF
  CASE ...                             && Reakcja na inne błędy
...
ENDCASE
...
```

Wydruk 2

W zasadzie każdy program musi mieć procedurę obsługi błędu (przynajmniej błędów 108 i 109), bowiem nawet komenda *USE* może wymagać reakcji na nieudaną próbę otwarcia pliku. Na wydruku 2 podano fragment przykładowej procedury obsługi błędu. Przedstawiony na nim sposób reakcji na udostępniane przez system błędy służy oczywiście bardziej celom demonstracyjnym niż użyciu w rzeczywistych programach. Ponadto, w procedurze obsługi błędu może być istotne oczekiwanie na dostęp do pliku lub na odblokowanie pliku bądź rekordu. Użycie tylko instrukcji *RETRY* może prowadzić do zakleszczenia (ang. *deadlock*). Przedstawione na wydruku 3 fragmenty programów mogą doprowadzić do zakleszczenia, jeśli plik *Odtw.prg* zawiera tylko instrukcję *RETRY*:

Węzeł 1	Węzeł 2
...	...
ON ERROR DO Odtw	ON ERROR DO Odtw
SELECT 1	SELECT 1
USE EXCLUSIVE Magazyn	USE EXCLUSIVE Kadry
SELECT 2	SELECT 2
USE EXCLUSIVE Kadry	USE EXCLUSIVE Magazyn
...	...

Wydruk 3

Na zakończenie warto przeanalizować, w jaki sposób w systemie dBase III Plus można uniknąć sytuacji prowadzących do niepoprawnego przetwarzania danych wymienionych na początku artykułu. Poniżej podano przykładowe rozwiązania.

1. Utrata wprowadzonej zmiany

```
P1:
  ...
DO WHILE. NOT. RLOCK( )
ENDDO
  REPLACE X WITH X+20
UNLOCK
  ...
P2:
  ...
DO WHILE .NOT. RLOCK( )
ENDDO
  REPLACE X WITH X+20
UNLOCK
  ...
```

2. Błędny odczyt

```
P1:
  ...
DO WHILE .NOT. RLOCK( )
ENDDO
  REPLACE X WITH X+50
  REPLACE Y WITH Y-50
UNLOCK
  ...
P2:
  ...
DO WHILE .NOT. RLOCK( )
ENDDO
```

```
  suma = X + Y
UNLOCK
```

3. Ponowny dostęp do danych, które uległy zmianie

```
P1:
  ...
DO WHILE. NOT. FLOCK( )
ENDDO
  KK = RECOUNT( )
  ... drukuj nagłówek
  GO TOP
  DO WHILE. NOT. EOF( )
    ... drukuj rekord
    SKIP
  ENDDO
UNLOCK
  ...
P2:
  ...
DO WHILE. NOT. FLOCK( )
ENDDO
  APPEND BLANK
  ... wypełnij pola
UNLOCK
  ...
```

Należy jednak zauważyć, że o poprawności danego rozwiązania decyduje właściwy sposób wykorzystania mechanizmów synchronizacyjnych przez użytkownika lub uruchomiony program.

Rozważmy fragment dwóch programów. Program pierwszy oblicza sumę pola *Wartość* pierwszego rekordu pliku *Rach-A* oraz pola *Wartość* pierwszego rekordu pliku *Rach-N*.

```
P1:
  ...
SET EXCLUSIVE OFF
USE Rach-A
DO WHILE .NOT. RLOCK( )
ENDDO
  suma = Wartość
USE Rach-N
DO WHILE .NOT. RLOCK( )
ENDDO
  suma = suma + Wartość
UNLOCK
  ...
```

Drugi program dokonuje przelewu całej wartości pola *Wartość* dla pierwszego rekordu pliku *Rach-N* do pierwszego rekordu pliku *Rach-A*. Poniżej podano dwie wersje programu. Pierwsza wersja może prowadzić do błędnego odczytu w programie *P1*, natomiast druga wersja do takiego błędu nie prowadzi.

P2 (wersja 1):

```
  ...
SET EXCLUSIVE OFF
SELECT 1
USE Rach-N
DO WHILE .NOT. RLOCK( )
ENDDO
  zw = Wartość
  REPLACE Wartość WITH 0
UNLOCK
SELECT 2
USE Rach-N
DO WHILE .NOT. RLOCK( )
ENDDO
  REPLACE Wartość WITH Wartość+zw
UNLOCK
  ...
```

P2 (wersja 2):

```
  ...
SET EXCLUSIVE OFF
SELECT 1
USE Rach-N
DO WHILE .NOT. RLOCK( )
ENDDO
SELECT 2
USE Rach-A
DO WHILE. NOT. RLOCK( )
ENDDO
SELECT 1
z_wart = Wartość
REPLACE Wartość WITH 0
UNLOCK
SELECT 2
REPLACE Wartość WITH Wartość+zw
UNLOCK
  ...
```

Zauważmy, że przelew jest w tym wypadku operacją niepodzielną – tzw. **transakcją** (ang. *transaction*) i dlatego wymaga wcześniejszego zablokowania obydwu rekordów przed wprowadzaniem jakichkolwiek zmian w ich polach.

PODSUMOWANIE

Rozpatrując dBase III Plus z punktu widzenia zastosowań w lokalnych sieciach komputerowych warto zwrócić uwagę, że system ten nie był tworzony z myślą wykorzystania go w tym środowisku, lecz został do niego przystosowany. Polegało to na wyeliminowaniu możliwości jednoczesnego, niekontrolowanego wprowadzania zmian do bazy danych z kilku węzłów i utworzeniu w ten sposób podstaw do poprawnego przetwarzania danych, bez czego żaden system w ogóle nie nadawałby się do użytku. Tworzenie bardziej wyrafinowanych algorytmów synchronizacji natomiast spada na użytkowników, do czego służą im jawne mechanizmy sterowania dostępem i blokowania. Takie rozwiązanie wraz ze sposobem jego realizacji (np. odczyty pustych wartości, brak transakcji, konieczność unikania zakleszczeń czy automatyczne blokowania przy odczycie) – sprawia, że dBase III Plus powinien znajdować zastosowanie w bardzo prostym przetwarzaniu danych, jak np. wprowadzanie danych z kilku węzłów i wydruk wspólnego raportu. Twórcy bardziej złożonego oprogramowania sieciowego będą korzystać zapewne z innych, lepszych narzędzi, jak XQL firmy Novell, lub użycia systemu DOS 3.10 i usług niższych warstw sieci.

Imprezy

KOMPUTERY '89

Wzorem lat ubiegłych Rada Wojewódzka NOT w Wałbrzychu organizuje IV Ogólnopolską wystawę – Targi sprzętu i oprogramowania komputerowego. Impreza odbędzie się w hali sportowo-widowiskowej Ośrodka Sportu i Rekreacji **w Wałbrzychu w dniach 17-19 maja 1989 r.**

Jednocześnie **w dniach 15-17 maja** zorganizowana zostanie w Wałbrzychu II Ogólnopolska konferencja pn. **Informatyka w handlu.**

Obu imprezom będą towarzyszyły liczne sympozja, seminaria, szkolenia i prelekcje.

Adres organizatorów:
Naczelna Organizacja Techniczna
Rada Wojewódzka w Wałbrzychu
58-300 Wałbrzych
tel. 235-69, 258-83; teleks 0745323

MIĘDZYNARODOWE TARGI WYNALAZKÓW

To kolejna forma szerokiego upowszechnienia i popularyzowania innowacji technicznych. Tegoroczne Targi, organizowane już po raz siódmy, będą miały po raz pierwszy charakter międzynarodowy. Odbędą się one **w dniach od 8 do 12 maja 1989 r.**

Wszystkie przedstawione na Targach rozwiązania objęte zostaną systemem promocji a zespoły specjalistów dokonają wyboru najwartościowszych rozwiązań do prezentowania ich na różnych wystawach w kraju i za granicą. Zostanie także rozstrzygnięty konkurs na najlepszy projekt wynalazczy.

Adres organizatorów:
Ośrodek Postępu Technicznego
ul. Buczka 1b, 40-955 Katowice
tel. 59-60-61 w. 294, 225

NIE DAJEMY RECEPT
SPRZEDAJEMY NARZĘDZIA

ELEKTRONIKA FILM KOMPUTER

Zakład Spółdzielni Pracy UNICUM

ul. Barska 3/20, 02-315 Warszawa
tel. 23-67-57, tlx 816955

OFERUJE USŁUGI W DZIEDZINIE KOMPUTERYZACJI PRZEDSIĘBIORSTW

Podejmuje się kompleksowej obsługi kontrahentów:

- **sprzedaż sprzętu mikrokomputerowego (kompletacja, dostawa, serwis gwarancyjny i pogwarancyjny)**
- **opracowanie oprogramowania użytkowego (wdrożenie, szkolenie personelu)**

Służymy Państwu:

- **doradztwem organizacyjnym**
- **projektowaniem, oprogramowaniem oraz wdrażaniem systemów dedykowanych dla konkretnego użytkownika**
- **opracowaniem unikalnych programów wraz z nadzorem autorskim**

Sprzedajemy profesjonalne narzędzia dla profesjonalistów.

EO/955/88

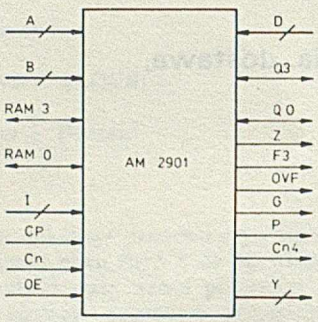
HENRYK GIZDOŃ
ADAM PAWLAK
WŁODZIMIERZ WRONA
Instytut Elektroniki
Politechnika Śląska
Gliwice

Zastosowanie języka VHDL do opisu i weryfikacji projektów układów cyfrowych

Artykuł ten zamyka cykl poświęcony językowi VHDL. W obecnej publikacji zamieszczono w miarę kompletny opis wybranego układu o dużym stopniu scalenia, co powinno się przyczynić do lepszego zrozumienia relacji zachodzących między różnymi jednostkami języka VHDL. Poruszono w niej także dotychczas nie omówiony mechanizm, tj. konfigurację ciała jednostki projektowej oraz weryfikację projektu przez symulację.

PROJEKT UKŁADU CYFROWEGO W JĘZYKU VHDL

Jako przykład prezentujący wykorzystanie języka VHDL przedstawiono opis układu scalonego Am2901, opracowanego przez firmę Advanced MicroDevices (rys. 1) [2]. Układ ten jest 4-bitowym segmentem ścieżki danych używanym do budowy mikroprocesorowych systemów segmentowych. System taki oprócz ścieżki danych zawiera część sterującą, realizowaną jako strukturę mikroprogramowaną. Podstawowymi elementami tej struktury są: układ sterujący, tzw. sekwenser, Am2909/2911 oraz pamięć mikroprogramu.



Rys. 1. Oznaczenie sygnałów wejściowych i wyjściowych układu Am2901

Układy rodziny Am2900 produkowane w technologii bipolarnej zaliczają się do układów o dużym stopniu scalenia LSI. Segment Am2901 stał się również przemysłowym standardem ścieżki danych, tzw. kompilatorów krzemowych ścieżki danych [6].

Deklarację sprzęgu jednostki projektowej Am2901 można zapisać w sposób przedstawiony na wydruku 1. W deklaracji tej zastosowano logikę wielowartościową pozwalającą modelować układy z wyjściami trójstanowymi. Deklaracje typów i funkcje związane z logiką wielowartościową oraz definicję atrybutu obciążalności wyjść portów sygnałów zawarto w pakiecie o nazwie *LOG_4WART* [3].

Deklaracja sprzęgu jednostki Am2901 oprócz deklaracji jego portów zawiera:
– deklarację parametru formalnego **generic** dla częstotliwości sygnału zegarowego,
– instrukcję komunikatu monitorującą wykrycie konfliktu na wyjściu *Y* (tzn. sytuacji, w której inna jednostka projektowa o wyjściu połączonym z wyjściem *Y* przyjmuje stan różny od stanu dużej impedancji, podczas gdy wyjście *Y* znajduje się w stanie aktywnym *'0'* lub *'1'*);
– klauzulę raportu wyprowadzającą informację o błędzie i klauzulę ważności komunikatu;
– specyfikację atrybutu obciążalności wyjścia *Y*.
Należy również zwrócić uwagę na typ fizykalny *FREQUENCY* występujący w deklaracji parametru formalnego, który musi zostać zdefiniowany przez projektanta, gdyż nie jest to typ zdefiniowany pierwotnie w języku VHDL.

```
with package LOG_4WART; use LOG_4WART;
entity Am2901
   (A, B: in BIT_VECTOR(3 downto 0);-- wejscia adresowe
    D: in BIT_VECTOR(3 downto 0);  -- dane wejsciowe
    -- trójstanowe wyjscie danych
    Y: out VECTOR_3STAN(3 downto 0);
    I: in  BIT_VECTOR(8 downto 0);  -- kod mikrooperacji
    -- sygnal wprowadzający wyjscie Y w stan wysokiej impedancji
    OE: in BIT;
    -- sygnały wyjściowe jednostki arytmetycznologicznej:
    -- przeniesienie, propagacja i generacja do układu szybkich
    -- przeniesień, nadmiar dla dodawania i odejmowania,
    -- najstarszy bit wyniku operacji, wynik operacji = "0000"
    Cn4, P, G, OVF, F3, Z: out BIT;
    Cn: in BIT;                    -- przeniesienie wejsciowe
    -- wejscie/wyjscie dla przesuwania w lewo/prawo komórki
    -- pamięci RAM i rejestru Q
    RAM3, RAM0, Q3, Q0: inout BIT_3STAN;
    CP: in BIT)                    -- sygnał zegarowy
    -- parametr formalny - częstotliwość zegara
    is generic
        (f: FREQUENCY:=10MHz)
    -- komunikat o błędzie w razie konfliktu na magistrali Y
    assert (Y/="EEEE")
      report
        "Konflikt na magistrali Y"
      severity ERROR;
    -- specyfikacja atrybutu obciążalności wyjscia Y
    for FANOUT of Y use 10;
end Am2901;
```

Wydruk 1

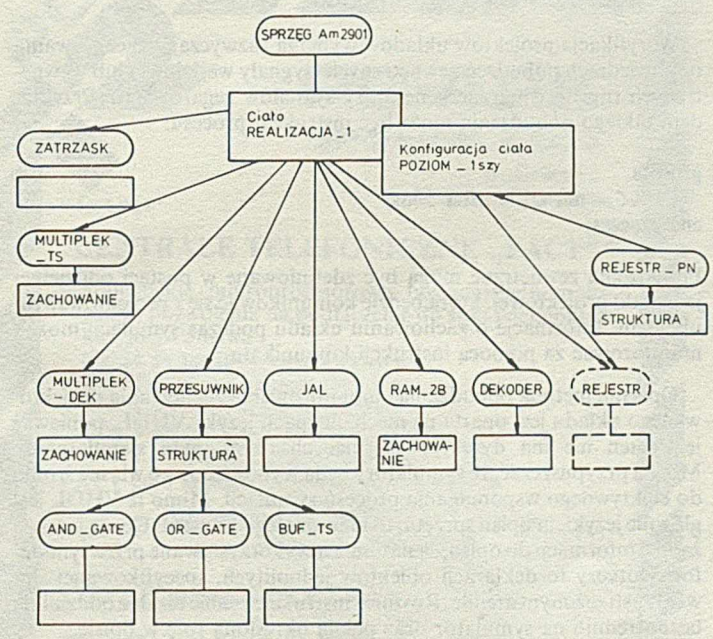
Struktura logiczna elementu Am2901 składa się z następujących modułów funkcjonalnych [2]: jednostki arytmetyczno-logicznej (JAL), grupy szesnastu rejestrów (pamięć dwuportowa RAM), multiplekserów źródła informacji wejściowej JAL, obiektu przeznaczenia informacji wyjściowej tej jednostki, bufora JAL (rejestr) oraz asynchronicznych układów przesuwania informacji. Układy te są połączone za pomocą wewnętrznych magistral czterobitowych oraz linii sterujących.

Definicję ciała jednostki projektowej Am2901 przedstawiono na wydruku 2. Definicja zawiera zarówno opis zachowania (w postaci instrukcji przypisania sygnałów *Z* i *F3*), jak i opis struktury. Pierwszym elementem definicji ciała są deklaracje sygnałów wewnętrznych: wejście pamięci *WE_P*, wejście *WE_R* oraz wejścia zatrzasków *WE_A* i *WE_B*. Następnie są zadeklarowane składniki struktury: multiplekser z wyjściem trójstanowym *MULTIPLEK_TS*, multiplekser dekoder *MULTIPLEK_DEK*, układ asynchronicznego przesuwania informacji *PRZESUWNIK* itd. Struktura połączeń składników została zapisana po słowie zastrzeżonym **begin**, za pomocą instrukcji konkretyzacji składników. Każdy składnik struktury jest reprezentowany przez nazwę, np. *PAM, ZAT_A, REJ_Q*. Definicje wybranych składników struktury jednostki projektowej Am2901 zostaną przedstawione w dalszej części artykułu, a elementy definicji pozostałych składników przedstawiono w [3].

```
architecture REALIZACJA_1 of Am2901 is
  block
    -- deklaracje sygnałów wewnętrznych:
    signal WE_P, WE_R, WE_A, WE_B: BIT_VECTOR(3 downto 0);
    signal WY_A, WY_B, Q, R, S, F: BIT_VECTOR(3 downto 0);
    signal EN_P, EN_R, SEL_Y, SHL, SHR, TRN: BIT;
    -- deklaracje składników struktury:
    component MULTIPLEK_TS port (A,B: in BIT_VECTOR;
      Y: out VECTOR_3STAN; S,OE: in BIT);
    component MULTIPLEK_DEK port (A,D,C,D: in BIT_VECTOR;
      Y,W: out BIT_VECTOR; SEL: in BIT_VECTOR(2 downto 0));
    component PRZESUWNIK  port (A,B: in BIT_VECTOR(3 downto 0);
      Y: out BIT_VECTOR(3 downto 0); SIO_L,SIO_R: inout BIT_3STAN;
      SHR,TRN,SHL: in BIT);
    component JAL  port (R,S: in BIT_VECTOR(3 downto 0);
      F: out BIT_VECTOR(3 downto 0); P,G,OVF,Cn4: out BIT;
      Cn: in BIT; OP: in BIT_VECTOR(2 downto 0));
    component ZATRZASK  port (D: in BIT_VECTOR;
      Q: out BIT_VECTOR(3 downto 0); G: in BIT);
    component REJESTR  port (D: in BIT_VECTOR;
      Q: out BIT_VECTOR; CP,EN: in BIT);
    component RAM_2B  port (AD_A,AD_B: in BIT_VECTOR(3 downto 0);
      D_WE: in BIT_VECTOR(3 downto 0);
      WY_A,WY_B: out BIT_VECTOR(3 downto 0); EN,W: in BIT);
    component DEKODER  port (ADR: in BIT_VECTOR(2 downto 0);
      EN_P,EN_R,SEL_Y,TRN,SHR,SHL: out BIT);
  begin
    -- generacja sygnału Z z wykorzystaniem instrukcji przypisania:
    Z <= not (F(0) or F(1) or F(2) or F(3)) after 10ns;
    F3<= F(3);
    -- utworzenie struktury połączeń:
    PRZES_P: PRZESUWNIK port (F,F,WE_P,RAM3,RAM0,SHR,TRN,SHL);
    PAM:     RAM_2B   port (A,B,WE_P,WE_A,WE_B,EN_P,CP);
    ZAT_A:   ZATRZASK port (WE_A,WY_A,CP);
    ZAT_B:   ZATRZASK port (WE_B,WY_B,CP);
    PRZES_R: PRZESUWNIK port (F,Q,WE_R,Q3,Q0,SHR,TRN,SHL);
    REJ_Q:   REJESTR  port (WE_R,Q,CP,EN_R);
    MULT_RS: MULTIPLEK_DEK port (D,WY_A,WY_B,Q,R,S,1(2 downto 0));
    JAL_F:   JAL      port (R,S,F,P,G,OVF,Cn4,Cn,1(5 downto 3));
    MULT_Y:  MULTIPLEK_TS port (WY_A,F,,),SEL_Y,OE);
    DEK_MP:  DEKODER  port (1(8 downto 6),EN_P,EN_R,SEL_),TRN,
                            SHR,SHL);
  end block;
end REALIZACJA_1;
```

Wydruk 2

Powyższy przykład posłuży do przedstawienia mechanizmu konfiguracji ciała jednostki projektowej, umożliwiającego zastąpienie niezdefiniowanego składnika innym, zdefiniowanym. Mechanizm ten pozwala również wybrać pożądaną definicję ciała jednostki projektowej, gdy z deklaracją sprzęgu tej jednostki jest związanych kilka definicji ciał.



Rys. 2. Hierarchia zagnieżdżania składników jednostki projektowej Am2901

Konfigurację ciała jednostki projektowej Am2901 o nazwie *POZIOM_1szy* dla rejestru *Q* przedstawiono na wydruku 3. Założono, że istnieje definicja innego rejestru (*REJESTR_PN*), który ma dodatkowe wyjścia zanegowane. Istotną częścią konfiguracji ciała jest lista połączeń portów. Lista ta może zawierać porty nie przyłączone, specyfikowane słowem zastrzeżonym **open**. Na rysunku 2 zilustrowano hierarchię zagnieżdżania składników jednostki projektowej Am2901.

```
configuration POZIOM_1szy of Am2901 for REALIZACJA_1 is
  for REJ_Q:REJESTR
    use
      entity (REJESTR_PN) -- rejestr z wyjściem prostym i zanegowanym
      port map (D=>D, Q=>Q, NQ=>open, CLK=>CP, EN=>EN)
      body (STRUKTURA)
    end for;
end POZIOM_1szy;
```

Wydruk 3

Konfiguracja ciała pozwala na wymianę składników struktury bez konieczności ciągłej modyfikacji definicji ciała, a zatem może to być definicja, w której postać pewnych składników zostanie zdefiniowana w końcowej fazie projektowania. We wcześniejszych fazach składniki te mogą być zastąpione innymi (już istniejącymi).

DEFINICJE SKŁADNIKÓW JEDNOSTKI PROJEKTOWEJ

Poniżej przedstawiono deklaracje sprzęgów i definicje ciał wybranych składników struktury jednostki projektowej Am2901, ilustrując wykorzystanie różnych instrukcji języka VHDL.

Multiplekser z wyjściem trójstanowym

Deklarację sprzęgu multipleksera z wyjściem trójstanowym przedstawiono na wydruku 4. Nie określono w niej uporządkowania i liczby bitów wektorów *A,B,Y*. Ustalenie tych cech jednostki projektowej nastąpi w instrukcji konkretyzacji składnika. Taka konwencja opisu składników umożliwia szersze ich wykorzystanie, nie tylko w opisie jednostki, dla której pierwotnie zostały zaprojektowane. Definicja ciała multipleksera opisująca jego zachowanie z użyciem instrukcji warunkowego przypisania sygnału przyjmuje postać przedstawioną na wydruku 5. W instrukcji przypisania zastosowano funkcję *ZAPISZ_WY* z pakietu *LOG_4WART* do transformacji logiki dwuwartościowej na wielowartościową. Funkcja *ZAPISZ_Z* ustala wektor o długości specyfikowanej atrybutem *RANGE*, którego każdy bit przyjmuje stan odpowiadający dużej impedancji. Użycie takiej konstrukcji jest niezbędne, ponieważ liczba bitów wektora *Y* nie jest z góry znana, co oznacza, że nie można napisać wprost: *Y<="ZZZZ"*.

```
with package LOG_4WART; use LOG_4WART;
entity MULTIPLEK_TS
  (A,B: in BIT_VECTOR;      -- wejścia informacyjne
   Y: out  VECTOR_3STAN;    -- wyjście trójstanowe
   S: in BIT;               -- wejście wyboru
   OE: in BIT)              -- OE='1' wprowadza wyjście
                            -- w stan wysokiej impedancji
  is
end MULTIPLEK_TS;
```

Wydruk 4

```
architecture ZACHOWANIE of MULTIPLEK_TS is
  block
  begin
    Y<=ZAPISZ_WY(A) after 10ns  when(S='0')and(OE='0') else
       ZAPISZ_WY(B) after 10ns  when(S='1')and(OE='0') else
       ZAPISZ_Z(Y'RANGE)    after 12ns;
  end block;
end ZACHOWANIE;
```

Wydruk 5

Multiplekser-dekoder

Utworzenie multipleksera-dekodera jako źródła informacji wejściowej jednostki arytmetyczno-logicznej wynika z dekompozycji funkcjo-

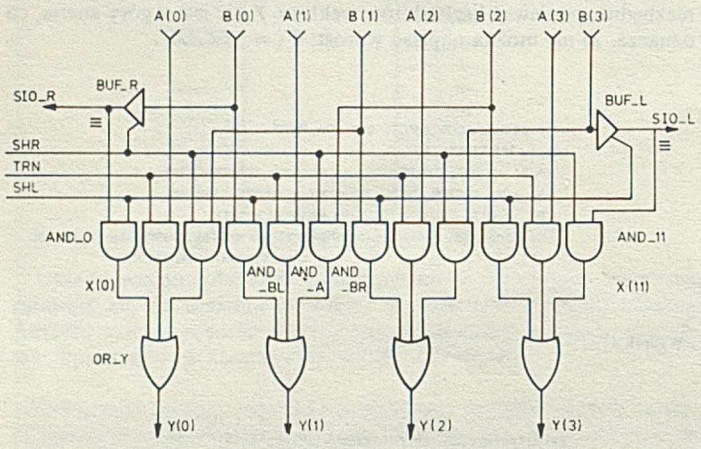
```
with function WART_NAT;
with function ZERO;
architecture ZACHOWANIE of MULTIPLEKSER is
  block
    -- deklaracja podtypu ZMIENNA_3BIT
    subtype ZMIENNA_3BIT is INTEGER range 0 to 7;
  begin
    process (A,B,C,D,SEL)
    begin
        case ZMIENNA_3BIT'(WART_NAT(SEL)) is
          when 0 => Y <= B; W <= D;
          when 1 => Y <= B; W <= C;
          when 2 => Y <= ZERO(Y'RANGE); W <= D;
          when 3 => Y <= ZERO(Y'RANGE); W <= C;
          when 4 => Y <= ZERO(Y'RANGE); W <= B;
          when 5 => Y <= A; W <= B;
          when 6 => Y <= A; W <= D;
          when 7 => Y <= A; W <= ZERO(W'RANGE);
        end case;
    end process;
  end block;
end ZACHOWANIE;
```

Wydruk 6

nalnej układu Am2901. Omówiona część struktury elementu Am2901 składa się z dwóch multiplekserów i dekodera trzybitowego pola kodu mikroinstrukcji. Zachowanie tej podstruktury zostało opisane w jednostce projektowej multiplekser-dekoder. Deklaracja sprzęgu tego składnika nie wnosi nowych elementów, zostanie więc pominięta. Definicję ciała multipleksera-dekodera można zapisać, jak na wydruku 6. Zachowanie multipleksera-dekodera opisano za pomocą instrukcji procesu z użyciem instrukcji **case**. W definicji ciała zastosowano funkcję *WART_NAT*,do wyznaczenia wartości całkowitej wektora bitów [3], oraz funkcję *ZERO* w analogicznym znaczeniu jak funkcję *ZAPISZ_Z*.

Układ asynchronicznego przesuwania informacji

Strukturę logiczną układu przesuwania informacji z wejść *B* o jeden bit w prawo i lewo oraz transmisji informacji nieprzesuniętej z wejść *A* na wyjścia *Y* przedstawiono na rys. 3. Ponieważ na tych samych końcówkach *SIO_L* i *SIO_R*, stanowiących wejścia, pojawia się sygnał wyjściowy, więc wyjścia szeregowe zrealizowano przez bufory trójstanowe. Deklarację sprzęgu jednostki projektowej przesuwnika można zapisać jak na wydruku 7. Strukturalną definicję ciała jednostki projektowej *PRZESUWNIK* przedstawiono na wydruku 8.



Rys. 3. Struktura logiczna asynchronicznego układu przesuwania informacji

```
with package LOG_4WART; use LOG_4WART;
entity PRZESUWNIK
  (A,B: in BIT_VECTOR(3 downto 0); -- informacja wejsciowa
   Y: out BIT_VECTOR(3 downto 0); -- informacja wyjsciowa
                                  -- A, ZB, B-Z
   SIO_L,SIO_R: inout BIT_3STAN; -- wejscie/wyjscie szeregowe dla
                                 -- przesuwania informacji w prawo-lewo
   SHR,TRN,SHL: in BIT)          -- sygnały sterujące: przesunięcie
                                 -- w prawo, transmisja, przesunięcie w lewo
  is
end PRZESUWNIK;
```

Wydruk 7

```
architecture STRUKTURA of PRZESUWNIK is
  block
    -- deklaracja sygnału wewnętrznego
    signal X:BIT_VECTOR(11 downto 0);
    -- deklaracje składników jednostki projektowej
    component AND_GATE port (A,B: in BIT; Y: out BIT);
    component OR_GATE  port (A,B,C: in BIT; Y: out BIT);
    component BUF_TS port (A: in BIT; Y: out BIT_3STAN;
                           OE: in BIT);

  begin
      BUF_R:BUF_TS port (B(0),SIO_R,SHR);
      BUF_L:BUF_TS port (B(3),SIO_L,SHL);
      for I:=0 to 3 generate
        AND_A:AND_GATE port (TRN, A(I), X(3*I+1));
        OR_Y:OR_GATE port (X(3*I),X(3*I+1),X(3*I+2),Y(I));
      end generate;
      for I:=0 to 3 generate
        if I:=0 generate
          AND_0 :AND_GATE port (CZYTAJ_WE(SIO_R),SHL,X(0));
          AND_11:AND_GATE port (SHR,CZYTAJ_WE(SIO_L),X(11));
        end generate;
        if I:=0 generate
          AND_BL:AND_GATE port (B(I-1),SHL,X(3*I));
          AND_BR:AND_GATE port (SHR,B(I),X(3*I-1));
        end generate;
      end generate;
  end block;
end STRUKTURA;
```

Wydruk 8

W definicji ciała użyto instrukcji powielania składników **generate**. Na uwagę zasługuje konieczność użycia funkcji *CZYTAJ_WE* do transformacji między sygnałami *SIO_L* i *SIO_R*, zdefiniowanymi w logice wielowartościowej, a sygnałami wejściowymi bramek *AND_0* i *AND_11*. W fizycznym układzie odpowiada jej połączenie bezpośrednie, ponieważ istnieje tylko jeden (ten sam) fizyczny sygnał. Użycie tej funkcji jest niezbędne do zapewnienia poprawności procesu symulacji.

WERYFIKACJA POPRAWNOŚCI PROJEKTU PRZEZ SYMULACJĘ

Prezentacja środków języka VHDL do projektowania układów cyfrowych wymaga przedstawienia metod badania poprawności projektów w tym języku. Podstawowe środowisko VHDL dostarcza do tego celu wielopoziomowy symulator. Autorzy niniejszych artykułów nie dysponowali symulatorem i kompilatorem języka VHDL. Chęć zaprezentowania pełnej koncepcji jego wykorzystania wymaga jednak dyskusji mechanizmów służących do symulacji.

Weryfikacja projektów układów wymaga zazwyczaj wygenerowania odpowiednich pobudzeń zewnętrznych (sygnały wejściowe) lub wewnętrznych (np. wewnętrzne generatory sygnałów zegarowych). Przykładem takiego pobudzenia może być instrukcja procesu:

```
process
  CLK<= not CLK after 50ns;
end process;
```

Pobudzenia zewnętrzne mogą być zdefiniowane w postaci oddzielnej jednostki projektowej, która będzie komunikować się z projektowanym układem. Informacje o zachowaniu układu podczas symulacji można monitorować za pomocą instrukcji komunikatu.

Opisana metoda pobudzania i monitorowania zachowania projektowanego układu jest oparta na mechanizmach języka VHDL, ponieważ język ten nie ma dyrektyw sterujących określonym symulatorem. Można przypuszczać, że symulatory będą wyposażone we własne środki do efektywnego wspomagania procesu symulacji. Mimo iż VHDL jest głównie językiem opisu sprzętu, to niektóre jego konstrukcje nie wnoszą żadnej informacji do opisu, są natomiast wykorzystywane przez symulator. Dotyczy to deklaracji obiektów jednolitych, specyfikowanej słowem zastrzeżonym **atomic**. Również instrukcje **enable disable** oddziałują bezpośrednio na symulator oraz pełnią określoną rolę w opisie.

Symulacja w języku VHDL ma charakter procesowy. Zapoczątkowanie symulacji wymaga określenia wartości początkowych wszystkich zmiennych, nośników i sygnałów. Początkowa wartość zmiennej może być definiowana w jej deklaracji lub przez dyrektywę inicjowania typu zmiennej. Początkowe wartości nośników sygnałów mogą być definio-

wane wyłącznie za pomocą dyrektyw inicjowania, a wartości sygnałów są obliczane z wartości swoich nośników.

Dyrektywa inicjowania składa się ze słowa zastrzeżonego **initialize**, po którym występuje nazwa typu i słowo zastrzeżone **to**, wskazujące wartość początkową. Dla nośników można opcjonalnie zapisać przyszłe wartości przebiegu czasowego. Przykładem dyrektyw inicjowania mogą być zapisy:

initialize ZMIENNA_38BIT **to** 7 ;
initialize BIT_3STAN **to** 'Z' ;
initialize DANE **to** B"0000", B"1111" **after** 100ns ;

Proces symulacji w języku VHDL przebiega cyklicznie. Symulacja rozpoczyna się w chwili uważanej umownie za zerową i kończy po upływie zadanego odcinka czasu ($\Delta > 0$), przy czym zawsze obejmuje przynajmniej dwa cykle. Każdy cykl symulacji składa się z następujących trzech kroków:

- wyliczenia aktualnych wartości wszystkich sygnałów, dla których nośniki zmieniły wartość;
- realizacji wszystkich instrukcji procesu, dla których wystąpiła zmiana wartości co najmniej jednego sygnału z umieszczonych na listach czułości poszczególnych procesów; w instrukcjach procesu mogą występować instrukcje przypisania przyporządkowujące nowe wartości nośnikom sygnałów;
- zwiększenie wartości wskaźnika czasu; globalny czas symulacji jest zwiększany do chwili, kiedy wartość przynajmniej jednego nośnika sygnału zmienia się.

Ponieważ przed rozpoczęciem symulacji wartości początkowe wszystkich sygnałów są już ustalone (obliczone z wartości nośników), więc w pierwszym kroku pierwszego cyklu symulacji nie występują sygnały, których nośniki zmieniły wartość. Jeśli występują instrukcje procesu z pustą listą czułości, tzn. realizowane zawsze niezależnie od zmian stanu jakiegokolwiek sygnału, to również drugi krok nie spowoduje zmiany stanu żadnego nośnika i symulacja zostanie zakończona. Dlatego w opisie projektu istotne są procesy z pustą listą czułości (np. zamieszczony powyżej proces generujący sygnał zegarowy), gdyż to one umożliwiają zapoczątkowanie symulacji.

* * *

VHDL jest pierwszym językiem opisu sprzętu, w którym wprowadzono koncepcję pakietu. Pakiet pozwala wyizolować z opisu zależności technologiczne, a także funkcje, deklaracje, tj. konstrukcje o charakterze uniwersalnym. Jednostka projektowa umożliwia reprezentację alternatywnych wersji projektu, co we wcześniejszych realizacjach języków opisu sprzętu nie było możliwe. Bardzo użyteczną cechą języka VHDL jest możliwość nadawania atrybutów jednostkom projektowym, sygnałom i portom. Za pomocą atrybutów można przekazywać informacje między innymi korzystając z istniejących stacji wspomagania procesu projektowania.

VHDL jest również pierwszym językiem opisu sprzętu umożliwiającym wyrażenie wielkości fizykalnych projektu w dokumentacji technicznej, dzięki wprowadzeniu typu fizykalnego wyrażającego miary wielkości fizycznych. Do opisu sprzętu można używać dwóch modeli opóźnień: modelu opóźnień transportowych (jak w prezentowanym przykładzie) oraz modelu inercyjnego umożliwiającego pochłanianie impulsów o czasie trwania krótszym niż zadeklarowany. Interesującą cechą języka VHDL jest możliwość wyrażania informacji o jednostkach projektowych o dowolnym charakterze, w postaci listy parametrów formalnych (generycznych).

VHDL ma również pewne ograniczenia, np.
- brak instrukcji zawieszania i wznawiania działania w opisie sekwencyjnym,
- brak możliwości definiowania obiektów dynamicznych,
- brak możliwości przesyłania informacji do (lub z) jednostek projektowych w postaci plików,
- brak możliwości przedefiniowania znaczenia (przeciążania) 23 podstawowych operatorów.

dokończenie na str. 29

WIESŁAW NOWIŃSKI
Instytut Podstaw Informatyki
Polska Akademia Nauk
Warszawa

Rekonstruowanie obrazów w tomografii komputerowej

Rekonstruowanie obrazów jest, obok rozpoznawania, dyskretyzacji, kompresji, segmentacji, jedną z dziedzin przetwarzania obrazów. Jego celem jest otrzymanie na podstawie danych pomiarowych (rzutów) zrekonstruowanego obrazu stanowiącego przybliżenie obrazu oryginalnego. Rekonstruowanie ma rozległy zakres zastosowań, np. w diagnostyce medycznej, radioastronomii, technice radarowej, czy mikroskopii elektronowej. Te tak różne zastosowania mają wspólne matematyczne i obliczeniowe podstawy. Największe jednak praktyczne wykorzystanie znalazło rekonstruowanie obrazów w tomografii komputerowej.

Celem niniejszego cyklu złożonego z trzech artykułów jest przedstawienie Czytelnikom problematyki z zakresu rekonstruowania obrazów w tomografii komputerowej wykorzystującej promienie *X*, a w szczególności wymagań na sprzęt komputerowy w tej dziedzinie, charakterystyki metod i algorytmów rekonstruowania obrazów oraz problemów implementacyjnych.

Niniejszy artykuł zawiera opis i charakterystykę poszczególnych elementów wchodzących w skład systemu rekonstruowania obrazów; dwa dalsze artykuły dotyczą algorytmów dla dwu podstawowych metod: rekonstruowania transformacyjnych i opartych na rozwinięciu w szereg.

TOMOGRAFIA KOMPUTEROWA

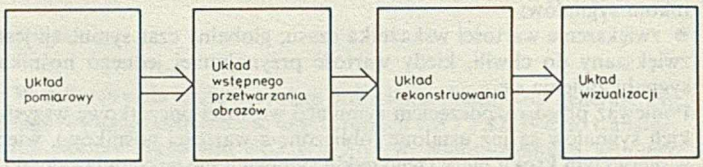
Po zbudowaniu pierwszego tomografu komputerowego przez Godfreya Hounsfielda i otrzymaniu przez niego w 1979 roku nagrody Nobla za prace w zakresie tomografii komputerowej, trwa niezwykle dynamiczny rozwój tej dziedziny. Jest on stymulowany nie tylko licznie prowadzonymi badaniami teoretycznymi, ale przede wszystkim udoskonaleniem technologii VLSI w zakresie przetwarzania obrazów oraz pracami nad architekturami przeznaczonymi do równoległego przetwarzania obrazów [3].

Celem tomografii komputerowej jest uzyskanie – za pomocą systemu komputerowego – zrekonstruowanego obrazu rozkładu gęstości na podstawie znajomości danych pomiarowych (rzutów), przy czym gęstość repezentuje rozpatrywaną właściwość badanego obiektu. W celu otrzymania danych pomiarowych stosuje się rozmaite techniki wykorzystujące różnorodne zjawiska. Zjawiska te są związane z procesami fizycznymi, umożliwiającymi badanie właściwości obiektów, np. za pomocą promieni X, promieni gamma, ultradźwięków, wiązek elektronów, prądów elektrycznych o małej częstotliwości lub też pól magnetycznych. Interpretacja gęstości zależy od stosowanej techniki pomiarowej. Przykładowo, dla promieni *X* gęstością jest współczynnik tłumienia tych promieni, dla promieni gamma – koncentracja substancji radioaktywnej wprowadzonej do obiektu, natomiast dla prądów elektrycznych o małej częstotliwości – przewodność.

Dr inż. WIESŁAW NOWIŃSKI ukończył w 1977 r. Wydział Elektroniki Politechniki Warszawskiej. Jest zastępcą kierownika Międzynarodowego Laboratorium Architektury Systemów Komputerowych przy Instytucie Podstaw Informatyki PAN oraz kierownikiem Zespołu Rekonstrukcji Obrazów. Specjalizuje się w zagadnieniach dotyczących cyfrowego przetwarzania obrazów i grafiki komputerowej, a w szczególności zajmuje się rekonstrukcją obrazów z rzutów w tomografii komputerowej.

SYSTEM REKONSTRUOWANIA OBRAZÓW

System rekonstruowania obrazów składa się z następujących podstawowych układów: pomiarowego, wstępnego przetwarzania obrazów, rekonstruowania i wizualizacji (rys. 1).



Rys. 1. Schemat systemu rekonstruowania obrazów

Zadaniem układu pomiarowego jest zbieranie rzutów obiektu, którego obraz (a dokładniej obraz pewnej cechy tego obiektu) należy zrekonstruować. Budowa tego układu zależy od wielu czynników, jak np. zjawiska fizycznego zastosowanego do badania obiektu, geometrii wiązki promieni, wymiarowości rzutów (jedno- lub dwuwymiarowe), współbieżności zbierania rzutów. Możliwości tego układu w znacznym stopniu wpływają na budowę i właściwości kolejnych układów systemu rekonstruowania obrazów. Przykładem może być układ DSR (ang. *dynamic spatial reconstructor*) [1, 7] zbierający jednocześnie 28 dwuwymiarowych rzutów. Układ DSR jest bardzo elastyczny, gdyż pozwala na różne konfiguracje pomiarowe w zależności od rozdzielczości czasowej i przestrzennej, grubości warstwy obiektu, kontrastu obrazu, stosowanych algorytmów rekonstruowania i technik zobrazowania.

Rzuty są obarczone błędami pomiarowymi, dlatego powinny być wstępnie przetworzone. Do tego celu służy układ wstępnego przetwarzania obrazów. Funkcję tego układu, jak i układów rekonstruowania oraz wizualizacji, może pełnić specjalizowany procesor. Architektury tego rodzaju procesora oraz występujące operacje były już prezentowane, m.in. w pracach [4, 5, 6].

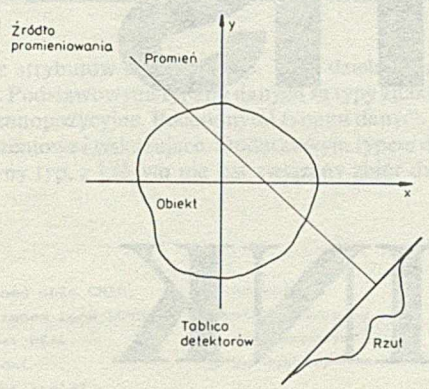
Budowa układu rekonstruowania jest zależna od zastosowanej metody i od wymiarowości wyniku rekonstrukcji. W rekonstruowaniu dwuwymiarowym, z jednowymiarowych rzutów rekonstruuje się dwuwymiarowy obraz. W rekonstruowaniu trójwymiarowym, z jednowymiarowych lub dwuwymiarowych rzutów, rekonstruuje się trójwymiarową scenę. W celu otrzymania trójwymiarowej sceny z jednowymiarowych rzutów, używa się sekwencji rekonstrukcji dwuwymiarowych dla kolejnych przekrojów obiektu równoległymi powierzchniami.

Rola układu wizualizacji obrazu nabiera szczególnej wagi przy rekonstruowaniu trójwymiarowym, gdy zrekonstruowana gęstość trójwymiarowego obiektu ma być przedstawiona na płaskim monitorze ekranowym. Wówczas istotnym zadaniem jest dobór właściwego zbioru operacji graficznych wykonywanych w trybie interakcyjnym.

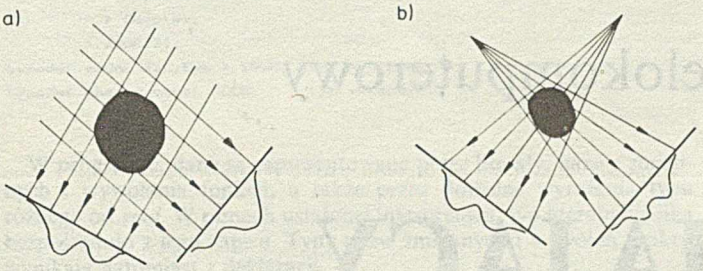
W rekonstruowaniu trójwymiarowym duże znaczenie ma również nie ujęta na schemacie przedstawionym na rys. 1 kompresja danych. Zauważmy, że trójwymiarowa scena składająca się z $256 \times 256 \times 256$ punktów o poziomach szarości kodowanych na jednym bajcie zajmuje 16 MB pamięci. Przesyłanie tak dużych obrazów, a zwłaszcza ich przechowywanie, staje się istotnym problemem.

UKŁAD POMIAROWY

Typowy sposób zbierania danych w tomografii komputerowej polega na wykonaniu dużej liczby pomiarów. Każdy pomiar jest związany określonym położeniem źródła promieniowania X i detektorów (rys. 2). Przy zbieraniu danych pomiarowych wykorzystuje się zjawisko usuwania części fotonów z wiązki emitowanej przez źródło promieniowania X w czasie przechodzenia fotonów przez badany materiał. Ponieważ prawdopodobieństwo p, że foton nie zostanie usunięty z promienia zależy od właściwości materiału, definiując współczynnik tłumienia materiału jako $\mu = -\ln\rho$ i przyjmując, że jest on gęstością, rekonstruuje się obraz f taki, że $f(x, y) = \mu(x, y)$.



Rys. 2. Zasada wykonywania pomiaru



Rys. 3. Geometrie wiązek promieni
a) równoległa, b) wachlarzowa

W rekonstruowaniu dwuwymiarowym stosuje się dwie geometrie wiązki promieni: równoległą i wachlarzową (rys. 3); ponadto, w rekonstruowaniu trójwymiarowym stosuje się wiązkę stożkową.

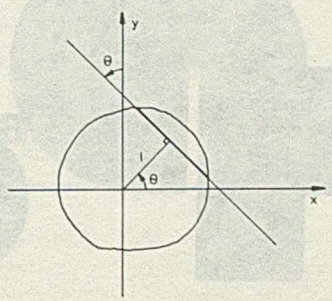
UKŁAD WIZUALIZACJI

Wynik rekonstrukcji dwuwymiarowej, czyli tablica wartości współczynnika tłumienia dla wszystkich punktów obrazu, jest przedstawiany w postaci graficznej (przedstawienie wyniku w postaci numerycznej jest nie do przyjęcia, gdyż rekonstruując obraz zdyskretyzowany, np. 256×256 punktów, otrzymuje się 64 K liczb, a dodatkowo z każdą z nich związane są dwie współrzędne punktu obrazu). Wskutek interakcji użytkownika z układem wizualizacji można wykonywać na obrazie operacje graficzne (przetwarzanie obrazu). Przykładowo, użytkownik może zmieniać skalę szarości, wykonywać detekcję brzegów i kształtów obiektów, wycinać obszary obrazu (np. obszary o zadanym poziomie szarości), powiększać fragmenty obrazu, nakładać obrazy, porównywać je, wygładzać brzegi obiektów czy też archiwować obrazy.

W celu zobrazowania trójwymiarowej sceny, na układ wizualizacji nakłada się dalsze wymagania umożliwiające wykonywanie takich operacji, jak zmiana układu obserwatora, transformacje geometryczne (obroty, przesunięcia, jednokładność), formowanie powierzchni obiektów, usuwanie niewidocznych powierzchni, podświetlanie czy też wybranie do wizualizacji dowolnego przekroju sceny.

SFORMUŁOWANIE PROBLEMU REKONSTRUOWANIA

Danymi wejściowymi w algorytmie rekonstruowania są tzw. sumy wzdłuż promienia, będące całkami wzdłuż prostych z funkcji obrazu f. Dokładniej mówiąc, wprowadza się pojęcie transformacji Radona R będącej całką z funkcji obrazu f wzdłuż prostej przechodzącej w odległości l od początku kartezjańskiego układu współrzędnych prostokątnych i przecinającej oś rzędnych pod kątem θ (rys. 4).



Rys. 4. Interpretacja transformacji Radona

Załóżmy znajomość przybliżeń y dokładnych wartości pomiarowych $p = Rf$ dla skończonego ciągu par $\{(l_i, \theta_i)\}$, $i = 1,...,I$ (wektor y jest tworzony ze wszystkich promieni dla wszystkich rzutów). Wówczas problem rekonstruowania formułuje się następująco:

dla danego wektora pomiarów y znaleźć przybliżenie $\hat{f}$ obrazu f.

Istnieje wiele sposobów wyznaczania wektora pomiarów: każdy z nich charakteryzuje się wnoszeniem różnych błędów pomiarowych i wymaga właściwego wstępnego przetwarzania, jak filtrowanie czy zwiększanie kontrastu. Podobnie, istnieje wiele sposobów rozwiązywania problemu rekonstruowania [2].

METODY REKONSTRUOWANIA

Istniejące metody rekonstruowania obrazów można zaklasyfikować do dwóch podstawowych grup: transformacyjnych metod rekonstruowania i metod rekonstruowania opartych na rozwinięciu w szereg.

W transformacyjnych metodach rekonstruowania wykorzystuje się zależności wyrażające obraz f w funkcji wektora pomiarów y, które to zależności należy w celu implementacji zdyskretyzować. Niech R^{-1} będzie odwrotną transformatą Radona. Wówczas $R^{-1}Rf = f$. Wobec tego znajomość dokładnych wartości danych pomiarowych $p = Rf$ umożliwia wyznaczenie obrazu f z zależności $f = R^{-1}p$. Innym podejściem umożliwiającym wyznaczenie obrazu f jest wykorzystanie bezpośrednich związków istniejących między transformatami Radona i Fouriera.

W metodach rekonstruowania opartych na rozwinięciu w szereg, problem rekonstruowania jest sformułowany od samego początku w postaci dyskretnej. Zakłada się, że obraz f można rozwinąć w szereg względem odpowiednio przyjętych obrazów bazowych. Prowadzi to do sformułowania dyskretnego problemu rekonstruowania, w którym poszukuje się wartości współczynników rozwinięcia w szereg. Dyskretny problem rekonstruowania sprowadza się do rozwiązywania dużych układów algebraicznych równań liniowych (istnieje również alternatywne sformułowanie problemu w postaci układu nierówności). W celu uniknięcia bezpośredniego rozwiązywania całego układu (np. metodą eliminacji Gaussa), co wymaga dużych mocy obliczeniowych, stosuje się różne techniki, jak np. dekomponowanie tego układu lub rozwiązywanie iteracyjne.

W artykule przedstawiono ogólny schemat systemu rekonstruowania obrazów wraz z opisem poszczególnych układów. Scharakteryzowano również podstawowe grupy metod rekonstruowania: metody transformacyjne i metody oparte na rozwinięciu w szereg. Mniejsze koszta (liczone jako czas pracy i niezbędna pojemność pamięci komputera) implementacji algorytmów opartych na metodach transformacyjnych powodują, że właśnie te algorytmy są powszechnie stosowane w dostępnych handlowo tomografach komputerowych.

dokończenie na s.29

JAN BIELECKI
Instytut Informatyki
Politechnika Warszawska

Turbo C

Jawne i niejawne konwersje danych

Jednym z atrybutów danej jest typ. Typy dzielą się na podstawowe i pochodne. Podstawowymi typami danych są typy znakowe, stałopozycyjne i zmiennopozycyjne. Pochodnymi typami danych są typy agregatowe, wyliczeniowe i wskazujące. Dodatkowym typem danych jest *void*. Jest to jedyny typ, z którym nie jest związany zbiór danych.

Przykłady

```
typedef signed char CHAR;          typ znakowy
typedef unsigned long LONG;        typ stałopozycyjny
typedef float REAL;                typ zmiennopozycyjny
typedef union{                     typ agregatowy
          int Arr[2];
          struct{
            float Re,Im;
          } Complex;
        } AGGR[3];
typedef enum{ Jan,Ewa } ENUM;      typ wyliczeniowy
typedef char * const ADDR;         typ wskazujacy
```

W programie, dane są reprezentowane przez literały, nazwy zmiennych i wywołania funkcji, a także przez dowolne wyrażenia typu różnego od *void*. W ramach ustalonej interpretacji, typ literału wynika bezpośrednio z jego zapisu. Typy nazw zmiennych i wywołań funkcji wynikają natomiast z deklaracji.

Przyjmuje się, że literały znakowe są typu *(int)*, a literały napisowe są typu *(char *)*. Typ literałów stałopozycyjnych zależy od przyjętej podstawy zapisu oraz tego, czy zostały zapisane z przyrostkiem czy bez przyrostka.

Z każdą postacią zapisu literału stałopozycyjnego jest związana uporządkowana lista typów, a z każdym typem jest związany zbiór danych. Uznaje się, że literał jest pewnego typu, jeśli reprezentuje daną należącą do zbioru związanego z tym typem. W wypadku wieloznaczności obiera się typ zajmujący na liście pozycję najwcześniejszą. A oto przyjęte listy typów:

- dla literału dziesiętnego bez przyrostka
(*int*)
(*long int*)
(*unsigned long int*)

- dla literału ósemkowego i szesnastkowego bez przyrostka
(*int*)
(*unsigned int*)
(*long int*)
(*unsigned long int*)

- dla literału z przyrostkiem *U* (albo *u*)
(*unsigned int*)
(*unsigned long int*)

- dla literału z przyrostkiem *L* (albo *l*)
(*long int*)
(*unsigned long int*)

- dla literału z przyrostkiem *U* i *L* (albo *u* i *l*)
(*unsigned long int*)

Należy podkreślić, że literały stałopozycyjne reprezentują dane o wartościach nieujemnych. Wynika stąd w szczególności, że napis –32768 nie jest literałem typu *(int)*, lecz że jest literałem typu *(long int)*.

Przykłady

```
Literał    Typ
'E'        (int)
"Ewa"      (char *)
45         (int)
32768      (long int)
0xFFF      (int)
0xFFFF     (unsigned int)
077U       (unsigned int)
45L        (long int)
0xCLu      (unsigned long int)
3.         (double)
.3f        (float)
3E2L       (long double)
```

Dana reprezentowana przez wyrażenie może być użyta bezpośrednio, albo może zostać poddana przekształceniu nazywanemu konwersją. Na wyróżnienie zasługuje konwersja *void*. Dana poddana takiemu przekształceniu zostaje odrzucona.

Przykład

```
void wait(void)
{
  while(!kbhit());
  (void)getch();
}
```

Wykonanie konwersji *(void)* powoduje odrzucenie danej udostępnionej przez funkcję *getch*.

Konwersje można podzielić na jawne i niejawne. Konwersje jawne są wyrażane za pomocą operatora konwersji, a konwersje niejawne są używane wszędzie tam, gdzie jest wymagana dana innego typu. Takie konwersje występują najczęściej podczas opracowania inicjalizacji i przypisań, podczas opracowania argumentów funkcji oraz podczas opracowania wyrażeń określających rezultat funkcji.

Przykłady

A. Konwersja podczas inicjalizacji

```
#include <stdio.h>
int Arr[] = { 2,3.0,(unsigned int)4 };
main()
{
  int Count = sizeof Arr / sizeof(int);
  while(Count--)
    printf(" %d", Arr[Count]);
}
```

Wykonanie programu powoduje wyprowadzenie liczb 4, 3 i 2. Podczas inicjalizacji następuje przekształcenie danej typu *(double)* i danej typu *(unsigned int)* na dane typu *(int)*.

B. Konwersje podczas przypisań

```
#include <stdio.h>
char Name[] = "Jan & Ewa";
main()
{
  void *Ref;
```

```
  char *Ptr;
  Ref = Name + 6;
  Ptr = Ref;
  printf("%s", Ptr);
}
```

Wykonanie programu powoduje wyprowadzenie napisu *Ewa*. Podczas opracowywania przypisania, dana typu (*char* *), reprezentowana przez wyrażenie *Name* + 6, jest przekształcana na daną typu *(void* *), a dana reprezentowana przez wyrażenie *(Ref)* jest przekształcana na daną typu *(char* *).

C. Konwersje podczas opracowywania argumentów

```
#include <stdio.h>
main()
{
  void sub(char);
  char Chr = 'b';
  sub('j');
  printf("%c", Chr);
}
void
sub(char Chr)
{
  printf("%c", Chr);
}
```

Wykonanie programu powoduje wyprowadzenie napisu *jb*. Podczas opracowania argumentu funkcji *sub*, dana typu *(int)* reprezentowana przez *'j'* jest przekształcana na daną typu *(char)*, a podczas opracowania drugiego argumentu funkcji *printf*, dana typu *(char)*, reprezentowana przez *Chr*, jest przekształcana na daną typu *(int)*.

D. Konwersje podczas opracowywania wyrażeń zawartych w instrukcjach powrotu

```
#include <stdio.h>
main()
{
  double real(int);
  printf("%.2f", real(3));
}
double
real(int Fix)
{
  return Fix + 2;
}
```

Wykonanie programu powoduje wyprowadzenie liczby 5.00. Podczas wykonywania instrukcji powrotu, dana typu *(int)*, reprezentowana przez wyrażenie *Fix* + 2, jest przekształcana na daną typu *(double)*.

Przypisanie nie jest jedyną operacją dwuargumentową, której opracowanie może wymagać dokonania konwersji. Warunkiem wykonania wielu innych operacji dwuargumentowych jest bowiem sprowadzenie obu argumentów do tego samego typu.
Jeśli okaże się to konieczne, są wówczas wykonywane przekształcenia nazywane typowymi konwersjami arytmetycznymi. Konwersje te są wykonywane w następującej kolejności:

- jeśli jeden argument jest typu *(long double)*, to drugi jest poddawany konwersji na typ *(long double)*;
- jeśli jeden argument jest typu *(double)*, to drugi jest poddawany konwersji na typ *(double)*;
- jeśli jeden argument jest typu *(float)*, to drugi jest poddawany konwersji na typ *(float)*;
- jeśli jeden argument jest typu *(unsigned long int)*, to drugi jest poddawany konwersji na typ *(unsigned long int)*;
- jeśli jeden argument jest typu *(long int)*, to drugi jest poddawany konwersji na typ *(long int)*;
- jeśli jakikolwiek argument jest typu *(char)* albo *(short int)*, ze znakiem albo bez znaku, to jest (bez zmiany wartości danej) poddawany konwersji na typ *(int)*, a jeśli nie jest to zawsze wykonalne, jest poddawany konwersji na typ *(unsigned int)*;
- jeśli jeden argument jest typu *(unsigned int)*, to drugi jest poddawany konwersji na typ *(unsigned int)*;
- jeśli okaże się, że żadna z opisanych konwersji nie jest wymagana, to oba argumenty są typu *(int)*.

Przykład

```
#include <stdio.h>
main()
{
  unsigned short int Fix = 0x8000;
  printf("%u", Fix);
}
```

Wykonanie programu powoduje wyprowadzenie liczby 32768. Drugi argument funkcji *printf* jest poddawany niejawnej konwersji na typ *(unsigned int)*.

Zasady wykonywania poszczególnych konwersji są takie same w przypadku konwersji jawnych i niejawnych. Ich skutki zależą jednak często od sposobu reprezentowania danych, a niekiedy i od założeń implementacyjnych. Z tego powodu zaleca się, aby złożone konwersje były zadawane jawnie. Taki sposób postępowania lepiej dokumentuje semantykę programu i zwiększa jego przenośność.

Nie wdając się w szczegółowe opisy można podać, że konwersje danych całkowitych *„krótszych"* na *„dłuższe"* odbywają się z zachowaniem wartości danej, a konwersje danych całkowitych *„dłuższych"* na *„krótsze"* zależą od tego, czy typ docelowy zawiera modyfikator *signed* czy *unsigned*. W pierwszym wypadku nie następuje zmiana wartości danej, chyba że jest to niewykonalne (kiedy to rezultat konwersji zależy od implementacji), a w drugim jest daną o wartości modulo *„liczba danych zbioru związanego z typem docelowym"*.

Uwaga. W implementacji Turbo C w obu przypadkach jest stosowana konwersja „modulo".

Pozostaje zauważyć, że jeśli dane ujemne są reprezentowane w systemie *U2*, to konwersje z typu *„ze znakiem"* na typ *„bez znaku"* i odwrotnie odbywają się bez zmiany reprezentacji danej. Jedynie w wypadku konwersji na typ *„dłuższy bez znaku"* następuje powielenie bitu znaku danej.

Przykład

```
#include <stdio.h>
main()
{
  signed char chrS = -1;
  unsigned int intU = (signed char)-1;
  printf("%d %u", chrS,intU);
}
```

Wykonanie programu powoduje wyprowadzenie liczb -1 i 65535.

Konwersje danych typów zmiennopozycyjnych na dane typów całkowitych polegają na odrzuceniu części ułamkowej wartości danej. Konwersje odwrotne mogą natomiast powodować pewną utratę dokładności. Utrata dokładności może nastąpić także podczas konwersji z typu zmiennopozycyjnego *„dłuższego"* na *„krótszy"*. Nie występuje ona natomiast w wypadku odwrotnym.

Przykład

```
#include <stdio.h>
main()
{
  signed char Chr = - 4.9;
  float Long = 123456789L;
  printf("%d %ld", Chr,(long)Long);
}
```

Wykonanie programu powoduje wyprowadzenie liczb -4 i 123456784 (sic!).

Sposób wykonywania konwersji danych całkowitych na dane wskazujące i odwrotnie oraz sposób wykonywania konwersji danych wskazujących na inne dane wskazujące, zależy od implementacji. Wymaga się jedynie, aby zawsze była wykonalna konwersja wskazania typu (*void**) na dowolne inne wskazanie, oraz aby konwersja danej wskazującej obiekt o większych wymaganiach co do sposobu rozmieszczenia w pamięci, na daną wskazującą obiekt o mniejszych takich wymaganiach i odwrotnie, nie powodowała zmiany wskazania. Przyjmuje się, że z punktu widzenia tej ostatniej pary konwersji, obszar pamięci wskazywany przez daną typu (*void* *) jest traktowany jak obiekt o najmniejszych wymaganiach.

Przykład

```
#include <stdio.h>
union{
  float Real;
  char  Char;
} Eqv;
main()
{
  Eqv.Real = 13.8;
  printf("%d", (int)*(float *)(char *)&Eqv.Real);
}
```

Wykonanie programu powoduje wyprowadzenie liczby 13. Wystąpi nawet w takiej implementacji, w której dane typu *(float)* wymagają, na przykład, rozmieszczenia na granicy słowa, a dane typu *(char)* mogą być rozmieszczane dowolnie.

W każdej implementacji jest dowolna jawna konwersja wskazania funkcji na wskazanie funkcji innego typu. Użyty operator konwersji może wówczas zawierać pseudoprototyp określający typy parametrów funkcji.

Przykład

```
#include <stdio.h>
main()
{
  void sub();
  ((void (*)(int))sub)(3.6);
}
void
sub(int Fix)
{
  printf("%d", Fix);
}
```

Wykonanie programu powoduje wyprowadzenie liczby 3. Użyta konwersja pełni w istocie rolę prototypu.

ANDRZEJ SIKORSKI
Warszawa

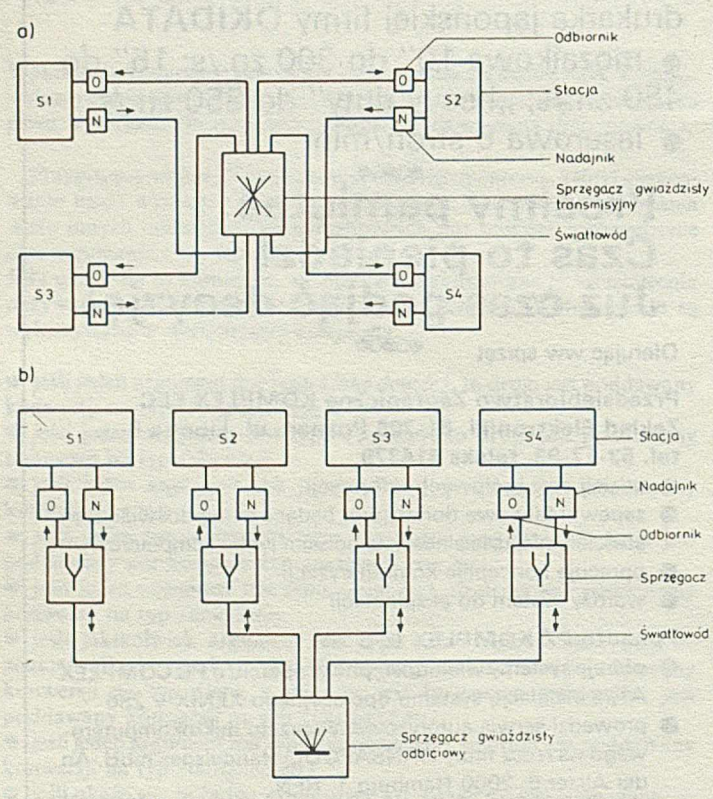
Lokalne sieci światłowodowe

Pojęciem lokalnej sieci komputerowej LAN określa się zbiór systemów komputerowych oraz innych urządzeń do przetwarzania, zapamiętywania, wprowadzania i wyprowadzania informacji w postaci cyfrowej, rozmieszczonych na ograniczonym obszarze, połączonych ze sobą za pomocą środków sprzętowych i programowych w sposób zapewniający szybką wymianę informacji. Sieci lokalne należą najczęściej do jednej instytucji i mają określone zadania do wykonania; obejmują na ogół urządzenia znajdujące się w jednym lub kilku sąsiadujących ze sobą budynkach, a zasięg ich nie przekracza zwykle obszaru o średnicy kilkunastu kilometrów.

Na sieć taką składają się: abonenci, stacje abonentów oraz podsieć komunikacyjna. Abonentami mogą być: komputery, systemy komputerowe, pamięci zewnętrzne, terminale, urządzenia sterowane numerycznie itd. Stacje abonentów są to urządzenia realizujące funkcje wspomagające w przekazywaniu informacji między abonentami przez podsieć komunikacyjną. Podsieć komunikacyjna zawiera łącza, które mogą być zbudowane z różnych ośrodków transmitujących informacje, takich jak: skręcona para przewodów, kabel współosiowy lub światłowód; ponadto podsieć zawiera dodatkowe urządzenia służące do przesyłania informacji.

Artykuł ten jest poświęcony sieciom lokalnym, w których ośrodkiem transmisji są światłowody. Transmisja danych za pomocą światłowodów ma następujące zalety:

- szerokie pasmo przenoszenia umożliwiające transmisję z dużą przepływnością binarną,
- możliwość przesyłania informacji na dość duże odległości bez pośrednich regeneratorów,
- odporność na zakłócenia elektromagnetyczne,
- praktyczne uniemożliwienie podsłuchu,
- elektryczna izolacja łączonych urządzeń.



Rys. 1. Światłowodowa sieć o konfiguracji gwiaździstej pasywnej ze sprzęgaczem: a) transmisyjnym, b) odbiciowym

Łącza światłowodowe można stosować w sieciach lokalnych o różnych rodzajach podsieci komunikacyjnych w różnych konfiguracjach i różnych metodach dostępu do łącza. Istnieją światłowodowe sieci lokalne o konfiguracjach: magistralowej, gwiaździstej, drzewiastej, pierścieniowej, a także o konfiguracjach mieszanych: stosujące zarówno deterministyczne, jak i przypadkowe metody dostępu do łącza.

Najprościej technikę światłowodową stosuje się w sieciach o tzw. konfiguracjach aktywnych, tzn. utworzonych z łączy dwupunktowych mających na swych końcach nadajnik i odbiornik. Są to sieci: gwiaździsta aktywna, drzewiasta aktywna i pierścieniowa aktywna [13]. Sieci takie umożliwiają również stosowanie techniki mieszanej, w której niektóre odcinki łącza są zrealizowane techniką światłowodową, a inne – przy użyciu kabla metalowego.

Sieci światłowodowe o tzw. konfiguracjach pasywnych mają bardziej złożone łącza, zawierające różnego rodzaju elementy optyczne, rozdzielające wiązkę promieniowania do różnych światłowodów [6, 12, 14]. Spośród sieci pasywnych stosunkowo prosto jest realizowana sieć o konfiguracji gwiaździstej pasywnej, logicznie identyczna z siecią magistralową. Elementem centralnym takiej sieci jest sprzęgacz gwiaździsty, który rozdziela promieniowanie przychodzące z nadajnika dowolnej stacji do odbiorników wszystkich dołączonych stacji. Sieci ze sprzęgaczem gwiaździstym transmisyjnym lub odbiciowym przedstawiono schematycznie na rysunku 1.

ROZWÓJ LOKALNYCH SIECI ŚWIATŁOWODOWYCH

Po raz pierwszy łącza światłowodowe zastosowano w lokalnych sieciach komputerowych w połowie lat siedemdziesiątych. Początkowo były to dwupunktowe łącza stosowane jako fragmenty sieci realizowanych z użyciem kabla metalowego. Jednym z pierwszych tego rodzaju zastosowań było połączenie dwukierunkowym łączem światłowodowym o długości 1,5 km dwóch fragmentów magistrali terminalowej, znajdujących się w dwóch budynkach w Houston (USA) [7].

Niedługo potem zaczęto budować kompletne sieci światłowodowe. Jeszcze w latach siedemdziesiątych zrealizowano w ośrodku badawczym firmy Xerox w Palo Alto światłowodową sieć komputerową Fibernet. Jest to światłowodowa wersja sieci Ethernet o konfiguracji magistralowej, zrealizowana w konfiguracji gwiaździstej pasywnej i umożliwiająca współpracę 19 stacji. Przy nie zmienionej organizacji logicznej sieci Ethernet zwiększono w ten sposób przepływność binarną z 10 do 114 Mb/s.

Na przełomie lat siedemdziesiątych i osiemdziesiątych w ośrodkach naukowo-badawczych różnych firm oraz w wyższych uczelniach powstało wiele różnorodnych eksperymentalnych sieci światłowodowych oraz sieci łączących technikę światłowodową z kablami metalowymi. Na przykład, w Uniwersytecie w Toronto (Kanada) powstała sieć Hubnet o konfiguracji drzewiastej i przepływności binarnej 50 Mb/s, zbudowana wspólnie z firmą Canstar [4]; w Uniwersytecie w Tsukuba (Japonia), przy współpracy firmy Mitsubishi Electric Co. zrealizowano sieć Gamma-net o konfiguracji pierścieniowej i przepływności binarnej 32 Mb/s [1].

Obecnie kilkadziesiąt firm oferuje sieci o rozmaitych konfiguracjach i bardzo zróżnicowanych parametrach [8, 11]. Podstawowe dane 23 wybranych sieci oferowanych przez 19 producentów zestawiono w tabeli 1. W sieciach tych stosuje się najróżniejsze metody dostępu do łącza, ale najczęściej jest realizowany dostęp rywalizacyjny CSMA/CD oraz dostęp z przekazywaniem uprawnienia (ang. *token*). Z pierwszą metodą dostępu wiąże się najczęściej gwiaździsta konfiguracja połączeń, z drugą – konfiguracja pierścieniowa. Najczęściej występuje konfiguracja pierścieniowa złożona z dwóch pierścieni.

Przepływności binarne w przedstawionych sieciach światłowodowych różnią się znacznie – od 1 do 180 Mb/s; najczęściej jednak znajdują się w zakresie 10–100 Mb/s.

W uniwersytetach i innych ośrodkach naukowo-badawczych powstają ciągle nowe rodzaje lokalnych sieci światłowodowych. Na potrzeby automatyki przemysłowej opracowano w General Motors i w AT&T światłowodowe wersje sieci MAP (ang. *manufacturing automation protocol*) [5]. Sieci MAP są sieciami o konfiguracji magistralowej i dostępie do łącza metodą przekazywania uprawnienia (ang. *token access*). W wersji dla małej liczby stacji i mniejszych odległości między stacjami sieci te mają konfigurację gwiaździstą pasywną, a rozmieszczone na większym terenie – mają konfigurację gwiaździstą aktywną.

Do zastosowań pomiarowych, zwłaszcza w reaktorach jądrowych opracowano w Japońskim Instytucie Energii Atomowej światłowodową wersję systemu CAMAC [9].

Najszybszą z istniejących obecnie sieci opracowano w uniwersytecie w Göteborgu przy współpracy California Institute of Technology, Pasadena (USA) [2]. Jest to sieć o konfiguracji pierścieniowej, łącząca trzy komputery WICAT łączem o przepływności binarnej do 3 Gb/s, wykonanym ze światłowodu jednomodowego. Źródłem promieniowania jest jednomodowa dioda laserowa zbudowana z InGaAsP, a detektorem fotodioda p-i-n z tranzystorem FED. Pracę sieci ograniczają obecnie do przepływności binarnej 250 Mb/s układy logiczne ECL, firmy Fairchild (najszybsze z dostępnych w handlu).

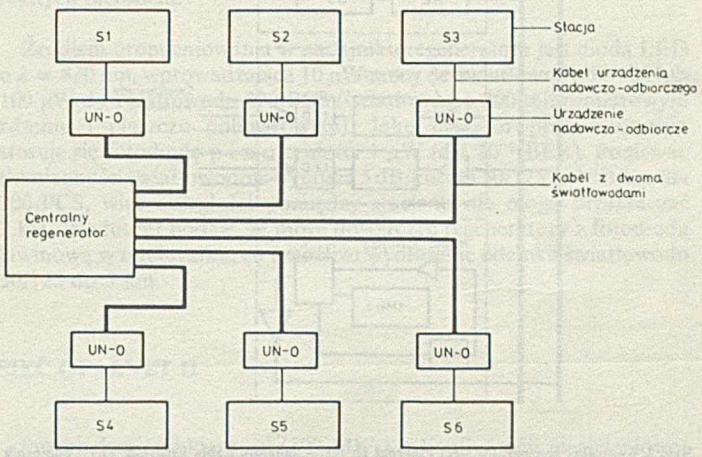
Na rozwój sieci światłowodowych w najbliższych latach będzie miało istotny wpływ uzgodnienie normy na szybką sieć światłowodową FDDI (ang. *fiber distributed data interface*) [3, 15], prace nad tą normą są już na ukończeniu.

FIBERNET II – ŚWIATŁOWODOWA WERSJA SIECI ETHERNET

Najpopularniejszą obecnie na świecie siecią LAN jest sieć Ethernet. Do początku 1986 r. zainstalowano ponad 10 000 takich sieci. Dla sieci tej firma Intel opracowała dwa układy LSI: 82501 i 82586, które wraz z układem nadawczo-odbiorczym IMDX 3015F realizują warstwy liniową i fizyczną stacji, zgodnie z normą ISO 8802.3. Umożliwia to budowę stacji taniej oraz o małych wymiarach.

Ethernet ma konfigurację magistralową i metodę dostępu do łącza CSMA/CD. Dane przesyła się kodem Manchester z przepływnością binarną 10 Mb/s. Dane są przesyłane w ramkach złożonych z początkowej części synchronizującej, adresu odbiorcy, adresu nadawcy, właściwej przesyłanej informacji i sum kontrolnych. Ośrodkiem transmisji jest kabel współosiowy o oporności falowej 50 Ω. Kabel składa się z 500-metrowych odcinków, do których można dołączyć 100 stacji; odcinki takie można łączyć przez regeneratory. Łącznie do sieci można dołączyć 1024 stacje, a maksymalna odległość między stacjami może wynosić 2,5 km. Ograniczenie odległości między stacjami wynika z tego, że czas propagacji sygnału tam i z powrotem musi być krótszy niż długość najkrótszej ramki, czyli – mniejszy od 51,2 ms.

Sieć Fibernet II [10], opracowana w ośrodku badawczym firmy Xerox, jest światłowodową wersją sieci Ethernet, w której zastąpiono kabel współosiowy wraz z urządzeniami nadawczo-odbiorczymi kablami światłowodowymi łączącymi wszystkie stacje z centralnym regeneratorem. Schemat tej sieci o konfiguracji gwiaździstej aktywnej przedstawiono na rys. 2.

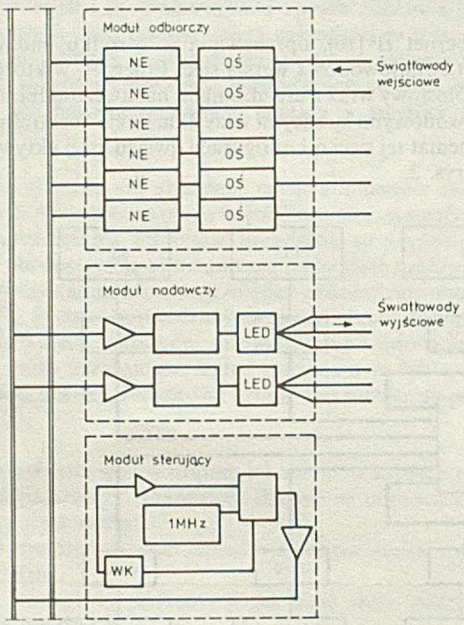


Rys. 2. Schemat sieci Fibernet II

Wybrane sieci światłowodowe

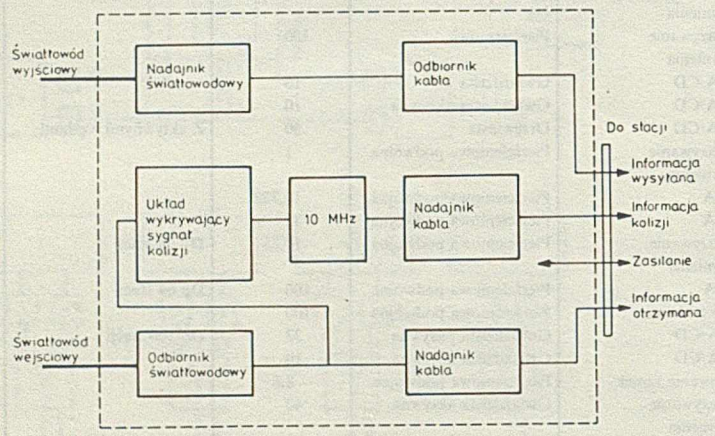
Producent	Kraj	Nazwa	Metoda dostępu	Konfiguracja	Szybkość transmisji Mb/s	Uwagi
Codenoll	USA	Codenet	CSMA/CD	Gwiaździsta pasywna	10	
Martin Marietta	USA	Filan	Wywołanie z priorytetami	Magistralowa	180	Częściowo – kabel metalowy
NEC Information Systems	USA	C & C-net loop	Przekazywanie uprawnienia	Pierścieniowa podwójna	32	Do 128 stacji
NEC	Japonia					
Proteon	USA	Pronet	Przekazywanie uprawnienia	Pierścieniowa	100	
Ungermann-Bass	USA	Net/One	CSMA/CD	Gwiaździsta	10	
Xerox	USA	Fibernet II	CSMA/CD	Gwiaździsta aktywna	10	
Canstar	Kanada	Hubnet	CSMA/CD	Drzewiasta	50	Z aktywnymi węzłami
Hitachi	Japonia	HLN 8140	Przekazywanie uprawnienia	Pierścieniowa podwójna	1	
		HLN 8380	TDMA	Pierścieniowa podwójna	10,752	
		Sigmanet	TDMA	Pierścieniowa podwójna	32	
Sumitomo Electric	Japonia	Suminet 3000	Przekazywanie uprawnienia	Pierścieniowa podwójna	10,25	Do 64 stacji
Toshiba	Japonia	Tosway	TDMA	Pierścieniowa podwójna	100	Do 64 stacji
NTT	Japonia	Opalnet II	Kombinowana	Pierścieniowa podwójna	100	
		Solarnet	CSMA/CD	Gwiaździsta pasywna	32	Do 100 stacji
BIIC	W. Brytania	Isolan	CSMA/CD	Gwiaździsta aktywna	10	
Focom	W. Brytania	1800 Series Data Net	Zajmowanie ramek	Pierścieniowa podwójna	8,6	
ICL	W. Brytania	Macrolan	Przekazywanie uprawnienia	Gwiaździsta aktywna	40	
Racal-Milgo	W. Brytania	Planet	Zajmowanie ramek	Pierścieniowa podwójna	10	Częściowo – kabel metalowy
Sension Communications	W. Brytania	Ethernet FO Repeater	CSMA/CD	Dwupunktowa	10	Łączy odcinki sieci Ethernet
Ericson	Szwecja	ZAM 10-1	Przekazywanie uprawnienia	Pierścieniowa	20	
GTE-ATEA	Belgia	Dateabus 232	Z rezerwacją czasu	Pierścieniowa	18,4	
		Dateabus 488	Z rezerwacją czasu	Pierścieniowa	12	
Hirschmann	RFN	Active Star	CSMA/CD	Drzewiasta	10	

W sieci tej centralny regenerator połączono z urządzeniami nadawczo-odbiorczymi kablem, zawierającym dwa światłowody gradientowe 100/140 (średnica rdzenia / średnica płaszcza światłowodu mierzonego w mikrometrach). W skład regeneratora wchodzą moduły odbiorcze, nadawcze oraz moduł sterujący, połączone dwoma odcinkami kabla współosiowego o oporności 50 Ω. Schemat regeneratora z jednym modułem odbiorczym i jednym modułem nadawczym pokazano na rys. 3. Sygnały odebrane przez dowolny z odbiorników modułu odbiorczego przesyła się kablem do modułu sterującego. Moduł sterujący zawiera układ wykrywający kolizję, który po jej wykryciu wysyła, przez drugi odcinek kabla i moduł nadawczy, do wszystkich urządzeń nadawczo-odbiorczych typowy sygnał kolizji sieci Ethernet o częstotliwości 1 MHz. Jeśli kolizja nie występuje, to sygnały przesłane do modułu sterującego rozsyła się tą samą drogą do wszystkich stacji.



Rys. 3. Centralny regenerator sieci Fibernet II; NE – nadajnik kabla Ethernet, OS – odbiornik światłowodowy, WK – układ wykrywania kolizji

Urządzenie nadawczo-odbiorcze połączono z centralnym regeneratorem kablem światłowodowym, a ze stacją – kablem wieloprzewodowym, typowym dla połączeń w sieci Ethernet. Urządzenie zawiera nadajnik światłowodowy z diodą LED (jako źródło promieniowania o długości fali λ = 850 nm), odbiornik z diodą p-i-n (jako detektor), układ wykrywający sygnał o wystąpieniu kolizji, nadajniki kabla i odbiornik kabla (rys. 4).



Rys. 4. Urządzenie nadawczo-odbiorcze sieci Fibernet II

W zrealizowanej wersji sieci zastosowano diodę LED emitującą promieniowanie o mocy 100 μW (-10 dBm), światłowód o tłumieniu 5 dB/km i detektor o czułości -28 dBm. Z bilansu energii otrzymano ograniczenie na długość kabla do 1250 m, co zapewnia maksymalną odległość między stacjami 2,5 km, identyczną jak w sieci Ethernet.

Po wymianie omówionych elementów maksymalna odległość między stacjami będzie ograniczona czasami propagacji sygnałów w sieci. Na podstawie analizy czasowej stwierdzono, że maksymalna długość kabla między centralnym regeneratorem a urządzeniem nadawczo-odbiorczym wynosi 2 km, tak więc odległość między stacjami nie może przekraczać 4,1 km (uwzględniając dwa 50-metrowe odcinki kabla między urządzeniami nadawczo-odbiorczymi a ich stacjami).

SIEĆ ACTIVE STAR

Innym przykładem sieci światłowodowej, opartej na zasadach działania sieci Ethernet, jest sieć Active Star firmy Hirschmann. W sieci tej łączy się elementy optyczne z elementami typowymi sieci Ethernet.

Podstawowymi elementami optycznymi sieci są: aktywny sprzęgacz gwiaździsty, optyczne urządzenia nadawczo-odbiorcze i kabel z dwoma światłowodami 50/125 lub 100/140.

Sprzęgacz jest urządzeniem, do którego można wstawić do 19 pakietów adapterów kabla światłowodowego lub adapterów kabla współosiowego. Informację przychodzącą na wejście jednego z adapterów przekazuje się na wyjścia wszystkich pozostałych adapterów. Adapter kabla światłowodowego zawiera światłowodowy nadajnik i odbiornik połączone z kablem dwoma złączami typu OSMA. Drugi koniec kabla połączono z urządzeniem odbiorczo-nadawczym lub z analogicznym adapterem innego sprzęgacza. Adapter kabla współosiowego zawiera nadajnik i odbiornik kabla oraz dwa gniazda do przyłączenia kabla współosiowego. Można do nich dołączyć dwa odcinki po 250 m kabla lub jeden odcinek długości 500 m.

Optyczne urządzenie nadawczo-odbiorcze wraz z kablem światłowodowym połączono ze stacją za pomocą typowego sprzęgu urządzeń nadawczo-odbiorczych sieci Ethernet. Urządzenie zawiera nadajnik i odbiornik identyczny jak w sprzęgaczu gwiaździstym oraz układ wykrywający kolizję i wysyłający sygnały o wystąpieniu kolizji.

W nadajnikach źródłem promieniowania jest dioda LED, emitująca 30 μW promieniowania o długości fali λ = 860 nm do światłowodu 50/125 lub 100 μW promieniowania do światłowodu 100/140.

W odbiornikach, jako detektora użyto fotodiody p-i-n o czułości 2 μW. Tak więc, dla kabla o tłumienności 5 dB/km maksymalna dopuszczalna długość kabla wynosi 3 km.

Sieć Active Star została zainstalowana w wielu instytucjach, między innymi w Uniwersytecie w Stuttgarcie, gdzie łączy komputery VAX rozmieszczone w sześciu budynkach. Odległość między najdalej od sieci umieszczonymi urządzeniami nadawczo-odbiorczymi wynosi około 2 km.

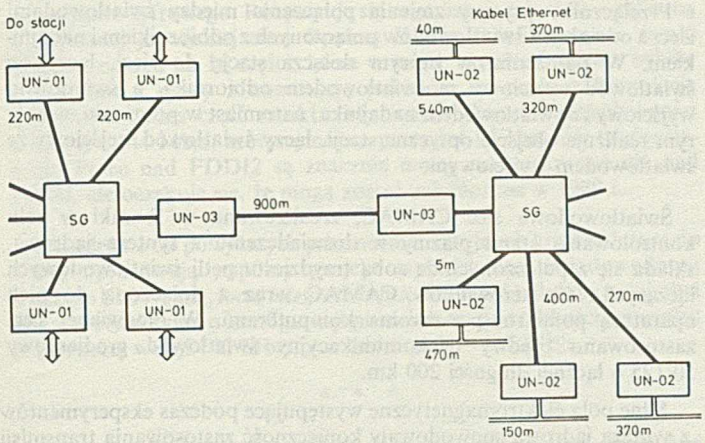
CODENET – SIEĆ GWIAŹDZISTA PASYWNA

Sieć Codenet (produkowana przez firmę Codenoll) jest, podobnie jak poprzednie sieci, światłowodowym rozszerzeniem sieci Ethernet. Jej podstawowymi elementami są pasywne sprzęgacze gwiaździste, światłowodowe urządzenia nadawczo-odbiorcze i łączące je kable światłowodowe.

Pasywne sprzęgacze gwiaździste oferowane przez producenta sieci mogą mieć w zależności od potrzeby 4, 8, 16, 32 lub 64 wejścia i wyjścia. Maksymalne straty mocy promieniowania powstałe podczas przejścia przez te sprzęgacze wynoszą odpowiednio: 10, 13, 17, 21 oraz 25 dB.

Urządzenia nadawczo-odbiorcze istnieją również w kilku wariantach. Wszystkie składają się z nadajnika i odbiornika światłowodowego, układu wykrywania kolizji i regulacji mocy emitowanego promieniowania. W różnych wersjach służą one do przyłączenia stacji, dołączenia odcinka kabla współosiowego lub jako regeneratory optyczne. W powyższych wersjach mają one układy sprzęgające łącza, nadajnik i odbiornik kabla lub drugą parę nadajnika i odbiornika światłowodowego. Urządzenia te produkuje się w różnych wersjach konstrukcyjnych: jako hybrydowy układ scalony, w postaci pakietu elektronicznego lub jako wolno stojące urządzenie.

Stosowane w sieci kable zawierają światłowody 100/140 o tłumienności 5 dB/km. Do połączeń wewnątrz budynków stosuje się kabel dwuświatłowodowy, a do połączeń między budynkami – kable czteroświatłowodowe (dwa zapasowe).

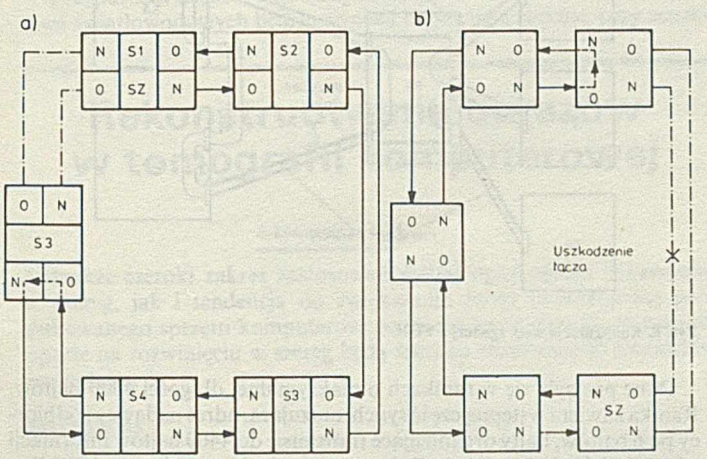


Rys. 5. Schemat jednej z instalacji sieci Codenet; SG – sprzęgacz gwiaździsty, UN-01 – urządzenie nadawczo-odbiorcze z interfejsem łącza, UN-02 – urządzenie nadawczo-odbiorcze z nadajnikiem kabla, UN-03 – urządzenie nadawczo-odbiorcze jako regenerator

Sieci Codenet są instalowane w USA i Europie od 1984 r. Na rysunku 5 przedstawiono schemat jednej z instalacji tej sieci, która łączy abonentów w siedmiu budynkach i składa się z dwóch sprzęgaczy gwiaździstych ośmiowejściowych, połączonych ze sobą kilkumetrowym odcinkiem kabla światłowodowego, będącego własnością poczty. Kabel ten łączy się ze sprzęgaczami przez regeneratory optyczne. Do jednego sprzęgacza dołączono cztery stacje rozmieszczone w dwóch budynkach, a do drugiego – pięć odcinków kabla współosiowego wraz z dalszymi stacjami. Odcinki te, umieszczone każdy w innym budynku, stanowiły uprzednio pięć niezależnych sieci Ethernet. Nie wykorzystane końcówki sprzęgaczy będą służyć do dalszej rozbudowy sieci.

SIEĆ 1800 SERIES DATA NET

Przykładem sieci o konfiguracji pierścieniowej i deterministycznej metodzie dostępu do łącza jest sieć 1800 Series (produkowana przez brytyjską firmę Focom). W sieci tej stacje są łączone dwoma pierścieniami łączy światłowodowych (rys. 6).



Rys. 6. Schemat sieci 1800 Series Data Net: a) konfiguracja początkowa, b) konfiguracja po uszkodzeniu łącza (SZ – stacja zarządzająca, N – nadajnik, O – odbiornik)

Jedna ze stacji zarządza pracą sieci. Generuje ona duże ramki 14,4 milisekundowe, składające się z 60 małych ramek, z których każda składa się z 256 ośmiobitowych szczelin (ang. *slot*). Z każdych ośmiu bitów tylko sześć służy do przesyłania informacji; także z 256 szczelin jedynie 240 służy do przesyłania informacji, a pozostałe 16 wykorzystuje się do sterowania pracą sieci.

Ramki z danymi regeneruje się w każdym regeneratorze kolejnej stacji i przesyła jednym pierścieniem łączy światłowodowych od pierwszej stacji (zarządzającej) do ostatniej, a drugim pierścieniem przesyła się je z powrotem. Połączenie ostatniej stacji z pierwszą nie jest wykorzystywane. W wypadku uszkodzenia któregoś z łączy, stacja sąsiadująca z uszkodzonym łączem automatycznie przejmuje funkcje stacji zarządzającej i sieć zmienia swą konfigurację (rys. 6b).

Dostęp do łącza jest określany przez stację zarządzającą wraz z tzw. monitorem, występującym w każdej stacji. Przydzielenie odpowiedniej szczeliny w każdej małej ramce do komunikacji między dwiema stacjami pozwala na przesyłanie między nimi, w obu kierunkach, 6 bitów co 240 μs, czyli realizację kanału 19,2 Kb/s (V24). Natomiast przydzielenie trzech szczelin umożliwia realizację kanału 76,8 Kb/s (V35). W ten sposób jest wykonywane w sieci przesłanie połączeniowe, nazywane w telekomunikacji komutacją kierunków.

Sieć może więc realizować jednocześnie 240 kanałów V24 lub 80 kanałów V35, albo odpowiednią ich kombinację. Przydzielenie kanałów odbywa się automatycznie w zależności od zgłaszanego zapotrzebowania. W pierścieniu można połączyć 30 stacji, z których każda może transmitować 16 niezależnych kanałów. Pierścienie można łączyć w jedną sieć przez połączenie elektryczne dwóch stacji, należących do różnych pierścieni.

Źródłem promieniowania w nadajniku regeneratora jest dioda LED o $\lambda = 820$ nm, wprowadzająca 10 μW mocy do światłowodu 50/125 lub 100 μW do światłowodu 200/PCS (światłowód o 200-mikrometrowym rdzeniu i płaszczu polimerowym). Jako detektor promieniowania stosuje się fotodiodę p-i-n o czułości 1 μW (dla 10^{-9} BER). Ponieważ tłumienności światłowodów wynoszą 5 dB/km dla 50/125 i 10 dB/km dla 200/PCS, więc odległości pomiędzy stacjami nie mogą przekraczać 2 km. Producent podaje, że może dostarczyć regeneratory z fotodiodą lawinową w odbiorniku, co umożliwi wydłużenie odcinka światłowodu 50/125 do 5 km.

SIEĆ OPALNET II

Przykładem szybkiej sieci (100 Mb/s) o konfiguracji pierścieniowej jest sieć Opalnet II opracowana w Japońskim Koncernie NTT [14]. W sieci Opalnet II, dzięki zastosowaniu specjalnej metody dostępu do łącza z podziałem czasu (TDMA), informację przesyła się jednocześnie metodą komutacji kierunków i metodą komutacji pakietów. Komutacja kierunków jest używana do przesyłania rozmów telefonicznych (64 Kb/s) i szybkiego faksymile (1,5 Mb/s), a komutacja pakietów – do przesyłania danych komputerowych. W sieci można utworzyć 480 kanałów o szybkości transmisji 64 Kb/s lub 20 kanałów o szybkości 1,5 Mb/s albo 90 Mb/s dla danych komputerowych.

Jest to sieć o konfiguracji pierścieniowej, w której stacje połączono podwójnym pierścieniem łączy światłowodowych. W sieci można połączyć 100 stacji. Sieć zawiera stację centralną zarządzającą pracą sieci. Stacja ta wysyła ramkę o długości 125 μs zawierającą początkową część synchronizującą, bity sterujące pracą sieci, bity opisujące rodzaj danych w poszczególnych grupach danych, grupy danych informacyjnych oraz ich bity kontrolne.

Poszczególne grupy danych w kolejnych ramkach albo są przyporządkowane jakiemuś połączeniu między stacjami i tworzą kanał informacyjny, albo mogą być zajmowane przez stację, która aktualnie posiada przekazane uprawnienie. Nie istnieje stały podział grup danych na oba rodzaje transmisji. Jeśli nie ma zapotrzebowania na przesyłanie z komutacją połączeń, to całą ramkę przeznacza się do transmisji danych komputerowych. W miarę zgłaszania zapotrzebowań pole ramki jest dzielone między oba rodzaje transmisji, lecz pewna część ramki zawsze zostaje dla transmisji z komutacją pakietów. Jednostka danych zajmowana w ramce może służyć do utworzenia jednego kanału o szybkości 1,5 Mb/s lub zmultipleksowanych 24 kanałów o szybkości 64 Kb/s.

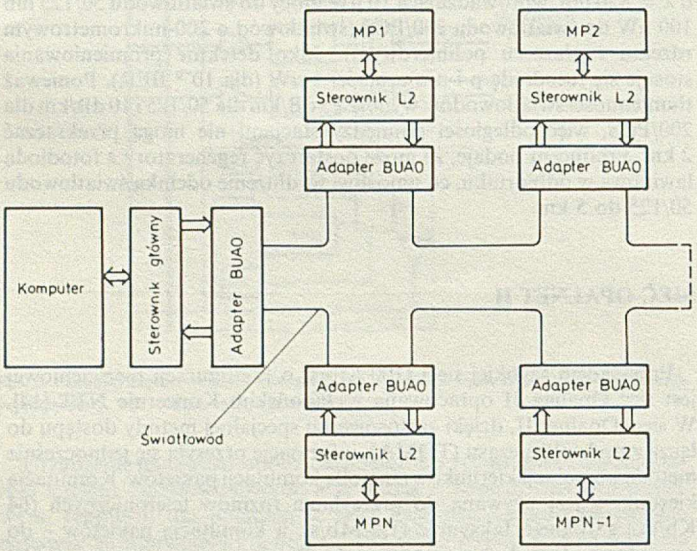
Światłowodowe łącza sieci Opalnet przesyłają informacje z szybkością (przepływnością binarną) 100 Mb/s kodem liniowym 8B/10B; światłowodem przesyła się więc impulsy promieniowania z częstotliwością 125 Mb/s. W łączach stosuje się światłowód 50/125 o tłumienności 3,5 dB/km. Jako źródła promieniowania używa się diody LED, a jako detektora – fotodiody lawinowej. Każda stacja ma obejście optyczne

zrealizowane za pomocą specjalnego sprzęgacza opracowanego w NTT, umożliwiającego przesyłanie danych do następnej stacji mimo wyłączenia lub uszkodzenia jednej ze stacji. Straty mocy promieniowania w sprzęgaczu podczas obchodzenia stacji wynoszą około 7 dB.

SIEĆ ŚWIATŁOWODOWA DLA SYSTEMU CAMAC

Magistrala CAMAC (ang. *computer automated measurement and control*) jest znanym systemem przesyłania danych służącym do komputerowego sterowania pomiarami. W systemie tym, do centralnej stacji z komputerem można przyłączyć siecią o konfiguracji pierścieniowej 62 stacje z urządzeniami pomiarowymi. Łączami wykonanymi z dziewięciożyłowego kabla przesyła się równolegle po 8 bitów z częstotliwością zegara od 100 KHz do 5 MHz.

W światłowodowej wersji sieci opracowanej w Instytucie Energii Atomowej w Ibaraki (Japonia; rys. 7) moduły pomiarowe sterowane standardowymi dla systemu CAMAC sterownikami L2 są połączone ze światłowodową pętlą przez moduły BUAO (ang. *byte serial U-port adapter for optical transmision*). Moduł BUAO dopasowuje szeregową transmisję światłowodową do równoległego sprzęgu systemu CAMAC. Moduł ten zawiera przełącznik optyczny, odbiornik oraz nadajnik światłowodowy, multiplekser i demultiplekser.



Rys. 7. Schemat światłowodowej wersji sieci CAMAC

Multiplekser zamienia 8 równolegle przesyłanych bitów na szeregowy ciąg impulsów. Zawiera zegar o częstotliwości 60 MHz generujący impulsy o czasie trwania około 8 ns. W celu przesłania 8 bitów jest generowany ciąg impulsów złożony z impulsu startu o potrójnej szerokości, impulsu synchronizującego i ośmiu impulsów informacyjnych. Okres między takimi grupami impulsów jest wypełniony impulsami dodatkowymi wysyłanymi z wypełnieniem 1/4.

Demultiplekser zamienia powyżej przedstawiony ciąg impulsów na 8-bitowy bajt. W demultiplekserze impuls synchronizujący przychodzący po impulsie startu wyzwala zegar o częstotliwości 60 MHz, który synchronizuje identyfikację impulsów informacyjnych bajtu. Po odebraniu wszystkich ośmiu bitów są one wraz z sygnałem zegarowym przesyłane równolegle do urządzeń CAMAC oraz do multipleksera w celu przesłania do dalszych stacji.

Nadajnik światłowodowy zamienia otrzymane z multipleksera impulsy na ciąg impulsów świetlnych wyprowadzonych do światłowodu. W zaproponowanym rozwiązaniu źródłem promieniowania jest dioda LED zbudowana z GaAlAs, emitująca promieniowanie o długości fali około 850 nm. Moc promieniowania wprowadzonego do światłowodu wynosi -22 dBm.

Odbiornik światłowodowy po zamianie sygnałów optycznych, otrzymanych ze światłowodu, na impulsy napięciowe i wzmocnieniu ich przesyła je do demultipleksera. Składa się on z krzemowej fotodiody lawinowej jako detektora, wzmacniacza i układu automatycznej regulacji wzmocnienia, który steruje zarówno wzmocnieniem wzmacniacza, jak i napięciem zasilania fotodiody. Zakres dynamiczny odbiornika wynosi od -22 do -40 dBm.

Przełącznik optyczny zmienia połączenia między światłowodami sieci a odcinkami światłowodów połączonych z odbiornikiem i nadajnikiem. W położeniu, w którym dołącza stację do sieci, łączy on światłowód wejściowy ze światłowodem odbiornika, a światłowód wyjściowy ze światłowodem nadajnika; natomiast w położeniu, w którym realizuje obejście optyczne stacji, łączy światłowód wejściowy ze światłowodem wyjściowym.

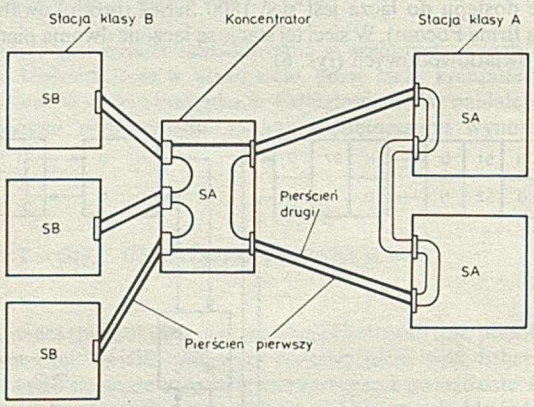
Światłowodowa sieć CAMAC, zrealizowana w Ibaraki w celu kontrolowania stanu plazmy w doświadczeniu z syntezą jądrową, składa się z połączonych ze sobą trzydziestu pętli światłowodowych łączących 160 sterowników CAMAC wraz z dołączoną do nich aparaturą pomiarową z dwoma komputerami. W omawianej sieci zastosowano typowy telekomunikacyjny światłowód gradientowy 50/125 o łącznej długości 200 km.

Silne pola elektromagnetyczne występujące podczas eksperymentów z syntezą jądrową spowodowały konieczność zastosowania transmisji światłowodowej.

FDDI – NOWY STANDARD SZYBKICH SIECI ŚWIATŁOWODOWYCH

Coraz bardziej staje się potrzbne utworzenie normy na szybką sieć światłowodową. Od kilku lat ponad 50 najpoważniejszych przedsiębiorstw (głównie amerykańskich) produkujących i stosujących sprzęt informatyczny opracowuje w ramach organizacji normalizacyjnej ANSI nową normę sieci światłowodowej o szybkości transmisji (przepływności binarnej) 100 Mb/s. Norma ta jest już bliska uzgodnienia i nosi nazwę FDDI (ang. *fiber distributed data interface* – sprzęg danych przesyłanych światłowodem).

Jest to sieć jednokanałowa o konfiguracji pierścieniowej i dostępie do łącza metodą przekazywania uprawnienia. W sieci wyróżnia się stacje klasy *A*, łączone między sobą dwoma pierścieniami, i stacje klasy *B*, dołączane tylko przez pierwszy pierścień do stacji klasy *A*. Przykład takiej konfiguracji pokazano na rys. 8.



Rys. 8. Konfiguracja sieci zgodnej z FDII

Dane przesyła się w ramkach o maksymalnej długości 4500 bajtów. Ramka zawiera wstępną część synchronizującą, adres nadawcy i odbiorcy po 6 bajtów, bajty organizujące transmisję do 4400 bajtów informacji oraz 8 bajtów kontrolnych. Dane są przesyłane z szybkością 100 Mb/s przy użyciu kodu liniowego 4B/5B, tak więc światłowodem przesyła się impulsy z przepływnością 125 Mb/s.

Nie dokonano jeszcze ostatecznego wyboru między światłowodami gradientowymi 62,5/125 a 85/125. Uzgodniono, że źródłem będzie dioda LED emitująca w paśmie 1300 nm, a detektorem – fotodioda p-i-n z tranzystorem FED. Oprócz parametrów łącza światłowodowego zatwierdzono protokoły warstw fizycznej i liniowej, a kilka firm półprzewodnikowych rozpoczęło opracowywanie układów scalonych realizujących te protokoły. Zaawansowane są także prace nad uzgodnieniem wyższych warst sieci.

Występująca potrzeba przesyłania tą samą siecią rozmów telefonicznych spowodowała powołanie grupy opracowującej drugą wersję sieci, nazwaną FDDI2. Sieć ta oprócz przesyłania danych, tak jak w sieci FDDI, tworzyłaby metodą komutacji kierunków kanały 64-Kbitowe dla telefonu cyfrowego. Rozpatruje się organizację takiej sieci zbliżoną do opisanej japońskiej sieci Opalnet II. Część informacyjna ramki byłaby dzielona na 16 części, które w miarę zapotrzebowania byłyby przełączane z pracy z komutacją pakietów na pracę z komutacją kierunków. Jedna część ramki realizowałaby kanał o przepływności binarnej 6,144 Mb/s, którym można przesyłać 96 kanałów telefonicznych. Prace nad FDDI2 są znacznie mniej zaawansowane niż nad FDDI, ale oczekuje się, że mogą zostać zakończone w 1989 r.

Niezależnie od powyższych prac normalizacyjnych kraje należące do EWG opracowują wspólny projekt szybkiej sieci światłowodowej o nazwie LION (ang. *local integrated optical network*). Byłaby to lokalna sieć wielofunkcyjna o konfiguracji pierścieniowej i szybkości transmisji w pierwszej wersji 140 Mb/s, a w późniejszej – 565 Mb/s.

* * *

Szacuje się, że w 1985 r. zainstalowano na świecie około 2600 sieci światłowodowych, a w 1986 r. około 3500, z tego w Stanach Zjednoczonych odpowiednio 1500 i 2000, a w Japonii – 700 i 1000. Pozostałe sieci występują na ogół w krajach europejskich, najwięcej w Wielkiej Brytanii. Powyższe liczby zainstalowanych sieci światłowodowych stanowią w przybliżeniu 10% ogólnej liczby zakładanych sieci lokalnych.

Przeważającą liczbę sieci światłowodowych przeznaczono do zastosowań biurowych i obliczeniowych, a jedynie niewielką ich część – do sterowania produkcją. Możliwości wytwarzania w dziedzinie sieci światłowodowych są obecnie zdecydowanie większe od istniejącego zapotrzebowania i to zarówno w zakresie liczby, jak i osiągalnych w realizacji parametrów technicznych. Tak więc to, ile i jakie lokalne sieci światłowodowe są obecnie instalowane, zależy wyłącznie od zastosowań tych sieci.

Wśród przyczyn, dla których realizuje się światłowodowe warianty sieci, na pierwszym miejscu znajduje się odporność sieci na zakłócenia elektromagnetyczne. W dalszej kolejności można wymienić: ochronę przesyłanej informacji przed podsłuchem, zbędność regeneratorów w dłuższych łączach oraz elektryczną izolację łączonych urządzeń. Szerokie pasmo przenoszenia nie jest tak istotne, gdyż obecne możliwości transmisyjne sieci Ethernet o szybkości transmisji 10 Mb/s są na ogół wykorzystywane w 5–15%.

Przypuszcza się, że w najbliższych latach wzrost liczby instalowanych sieci światłowodowych będzie wynosił około 50% rocznie, przy oczekiwanym wzroście wszystkich instalowanych sieci o około 40%. Szczególnie przewiduje się wzrost liczby instalowanych sieci o technice mieszanej, łączącej światłowody z kablami metalowymi.

Jak dotychczas, najczęściej instalowano sieci o rywalizacyjnym dostępie do łącza i konfiguracji gwiaździstej. Należy się spodziewać, że wzrośnie zainteresowanie sieciami o konfiguracji pierścieniowej i o różnych metodach dostępu z podziałem czasu (TDMA).

Nowe zastosowania sieci, takie jak przesyłanie dużych bloków danych między szybkimi komputerami oraz obsługa kolorowych terminali graficznych, zwiększy zapotrzebowanie na sieci o dużej (tzn. co najmniej 100 Mb/s) szybkości transmisji.

Lokalne sieci komputerowe rozwijają się też w kierunku sieci wielofunkcyjnych. Istnieje już wiele sieci lokalnych przesyłających zarówno dane komputerowe, jak i rozmowy telefoniczne oraz faksymile. Projektuje się rozszerzenie tych możliwości sieci także o przesyłanie wizji. Takie wielofunkcyjne sieci lokalne będą musiały spełniać normy wielofunkcyjnych sieci telekomunikacyjnych ISDN (ang. *integrated services digital network*), aby przez łącza telekomunikacyjne mogły współpracować z innymi sieciami.

LITERATURA

[1] Ebihara Y. et al.: Gamma-net – A Local Computer Network Coupled by a High Speed Optical Fiber Ring Bus. Computer Networks, No. 7, 1983
[2] Eng S. T. et al.: A 250-Mbit/s Ring Local Computer Network Using 1,3-μm Single-Mode Optical Fibers. Journal of Lightwave Techn., No. 3, 1985
[3] Finnie G.: Big Vendors Back Fibre Standard. Communication Systems Worldwide, No. 2, 1987
[4] Ikeman H., Lee E. S., Boulton P. I. P.: High-Speed Network Uses Fiber Optics. Electronics Week, No. 28, 1984
[5] Iversen W. R.: The Next Step for MAP – Fiber-Optic Networks. Electronics, No. 32, 1986
[6] Matsushita S., Kawai K., Uchita H.: Fiber-Optic Devices for Local Area Network Application. Journal of Lightwave Techn., No. 3, 1985
[7] McDermott J.: They're here – Interconnections for Computers and Instrumentation. Electronic Design, No. 22, 1978
[8] Minowa J., Tokura N., Nosu K.: Development of Fiber-Optic Local Area Networks in Japan. Journal of Lightwave Techn., No. 3, 1985
[9] Nishitani T. et al.: Optical Serial Highway for CAMAC System Using High-Performance Optical Bus Adapters. Journal of Lightwave Techn., No. 3. 1985
[10] Rawson E. G.: The Fibernet II Ethernet – Compatible Fiber-Optic LAN. Journal of Lightwave Techn., No. 3, 1985
[11] Rosenberger D., Witte H.H.: Optical LAN Activities in Europe. Journal of Lightwave Techn., No. 3. 1985
[12] Sikorski A.: Technika światłowodowa w lokalnych sieciach komputerowych. Techniki Komputerowe, nr 5–6, 1987
[13] Storozum S. L.: Fiber Optic Systems Practical Design. Photonics Spectra, No. 9–11, 1985
[14] Tokura N., Oikawa Y., Kimura Y.: High Reliability 100 Mbit/s Optical Accesing Loop Network Systems Opalnet-II. Journal of Lightwave Techn., No. 3, 1985.

Rekonstruowanie obrazów w tomografii komputerowej

dokończenie ze s. 19

Jednakże szeroki zakres zastosowań metod opartych na rozwinięciu w szereg, jak i tendencja do zwiększania mocy obliczeniowej produkowanego sprzętu komputerowego, mogą spowodować, że metody oparte na rozwinięciu w szereg będą szeroko stosowane w niedalekiej przyszłości.

LITERATURA

[1] Gilbert B.K. et al.: Ultra high-speed transaxial image reconstruction of the heart, lungs and circulation via numerical approximation methods and optimized processor architekture. Comput. Biomed. Research, Vol. 12, 1978
[2] Nowiński W.L.: Metody rekonstrukcji obrazów w tomografii komputerowej . Prace IPI PAN, nr 597, Warszawa, 1986
[3] Offen R.J. (Ed.): VLSI Image Processing. Collins, London, 1985
[4] Owczarczyk J.: Interakcyjny system przetwarzania obrazów. Informatyka, nr 2, 1984
[5] Owczarczyk J., Stolarski M., Woźniak E.: Architektura współbieżnych systemów przetwarzania obrazów. Informatyka, nr 5, 1986
[6] Owczarczyk J., Stolarski M., Woźniak E.: Współbieżne systemy przetwarzania obrazów – procesory tablicowe. Informatyka, nr 6, 1986
[7] Ritman E. L. et al.: Physics and technical consideration in the design of the DSR – a high temporal resolution volume scanner. AJR No. 134, 1980.

Zastosowanie języka VHDL

dokończenie ze s. 17

Bardziej wyczerpującą dyskusję wad i zalet języka VHDL można znaleźć w [5]. W prezentowanych artykułach autorzy opierali się na wersji 7.2 języka VHDL, będącej wyjściową do prac nad standardowym językiem opisu sprzętu, prowadzonych w Instytucie Inżynierów Elektryków i Elektroników (IEEE) w USA. Prace te zakończono w połowie roku 1987 definicją języka standardowego opartego na języku VHDL, eliminującą wiele wyżej wymienionych wad.

LITERATURA

[1] Aylor J.H., et al.: VHDL-Feature Description and Analys's. IEEE Design & Test, April 1986
[2] Bipolar μProcessor Logic. The data interface book, Advanced Micro Devices Inc., 1981
[3] Gizdoń H., Pawlak A., Wrona W.: Język opisu sprzętu VHDL – podstawowe mechanizmy. Informatyka, nr 11–12, 1988
[4] Gizdoń H., Pawlak A., Wrona W.: VHDL – Ada języków opisu i projektowania sprzętu. Informatyka, nr 10, 1988
[5] Nash J.D., Saunders L.F.: VHDL Critique. IEEE Design & Test, April 1986
[6] Romeliotis M., Armstrong I.R.: HDL Modeling for Process Oriented Simulation. Computer Hardware Description Languages and their Applications, IFIP, 1987
[7] Rowson J., Walker B.: Inside a 2901 Datapath Compiler. VLSI Systems Design, May 1986
[8] VHDL Language Reference Manual. Version 7.2. Report IR-MD-045-2, 1 August 1985
[9] VHDL Users's Manual, vol. 1-Tutorial. Report IR-MD-065-1, 1 August 1985.

Ceny ogłoszeń

Od 1 stycznia 1989 r. obowiązują następujące ceny materiałów reklamowych publikowanych na łamach INFORMATYKI:

Ogłoszenia czarno-białe, artykuły reklamowe i informacje naukowo-techniczne (biuletyny) zależnie od objętości: cała strona – 70 tys., 3/4 s. – 60 tys., 2/3 s. – 55 tys., 1/2 s. – 50 tys., 1/3 s. – 45 tys., 1/4 s. – 40 tys., 1/8 s. – 30 tys., poniżej 1/8 s. – 200 zł za cm².

Dodatki do ceny podstawowej – za każdy dodatkowy kolor + 30%, za każdy specjalny kolor (nie wynikający z podstawowych kolorów) + 30%, za pełny kolor (grafika wielobarwna, zdjęcia kolorowe) + 120%, za zamieszczenie ogłoszenia na I lub IV stronie okładki + 100%, za zamieszczenie ogłoszenia na II i III stronie okładki + 50%.

Zniżki dotyczą ogłoszeń – całkowitych powtórzeń: – za ogłoszenia 3-5-krotne – 10%, za ogłoszenia 6-10-krotne – 20%, za ogłoszenia 11-krotne i powyżej – 30%, za artykuły i wkładki reklamowe wykonywane przez zleceniodawcę – 40%, za biuletyny i bloki reklamowe – 60%. W innych uzasadnionych wypadkach dopuszcza się stosowanie rabatów specjalnych.

Ceny nadbitek reklamowych: wkładka 2 s. A4 – nakład do 500 egz. – 30 tys. zł, za każde następne 500 egz. – 25 tys. zł; wkładka 4 s. A4 – nakład do 500 egz. – 60 tys. zł, za każde następne 500 egz. – 50 tys. zł.

Ceny usług nie wymienionych w niniejszym cenniku będą ustalane każdorazowo na podstawie kalkulacji, jako ceny umowne. W wypadku rezygnacji Zleceniodawcy z wykonania zamówienia przed przekazaniem materiałów do druku – ponosi on koszty w wysokości 25% wartości zlecenia. W wypadku rezygnacji – gdy materiały są już w druku – Zleceniodawca ponosi pełne koszty ogłoszenia.
Niniejszy cennik dotyczy wyłącznie ogłoszeń firm krajowych i obowiązuje od 1 stycznia 1989 r. Ogłoszenia przyjęte przed tym terminem będą rozliczane według dotychczas obowiązującego cennika.

Ogłoszenia przyjmowane są przez:
Dział Ogłoszeń i Reklamy WCiKT NOT SIGMA,
ul. Świętojerska 5/7, 00-236 Warszawa
adres do korespondencji: skrytka pocztowa 1004, 00-960 Warszawa
telefony: 31-93-65 lub 31-22-21 w. 196 i 291

Dynamiczne tworzenie zadań w Adzie

W celu zilustrowania mechanizmu użycia typów zadaniowych i typów wskaźnikowych do dynamicznego tworzenia zadań w Adzie można posłużyć się współbieżną wersją algorytmu generowania liczb pierwszych. Zastosowany algorytm jest następujący:
1. Najmniejszą liczbą pierwszą jest 2.
2. Każda liczba naturalna $N>2$ jest liczbą pierwszą, jeżeli nie jest podzielna przez żadną liczbę pierwszą mniejszą od N.

Prosta realizacja tego algorytmu polega na utworzeniu dla każdej liczby pierwszej oddzielnego procesu, który sprawdza podzielność kolejnych liczb naturalnych przez tę liczbę pierwszą. Tak powstałe procesy można ustawić w ciąg, podobny do potoku w Unixie, i przepuszczać przezeń kolejne liczby naturalne w ten sposób, że liczba wykazująca niepodzielność przez i-ty element ciągu jest przekazywana do elementu $(i+1)$-ego, a dla liczby wykazującej niepodzielność przez wszystkie n elementów ciągu tworzy się nowy proces będący $(n+1)$-szym elementem (ponieważ jest to liczba pierwsza).

Przede wszystkim należy zdefiniować zadanie, które dostarcza liczb do potoku. Nazwano je *FEEDER*, a w jego ciele zawarto instrukcje: wprowadzania granicznej liczby obliczeń, tworzenie pierwszego zadania w potoku, odpowiadającego liczbie 2, i zasilania potoku kolejnymi liczbami naturalnymi (wydruk 1).

```
with IIO,TEXT_IO; use IIO,TEXT_IO;
procedure G_PRIMES is

        task FEEDER;

        task type CHECKER is
                entry Who_Am_I(My_Prime : in POSITIVE);
                entry Check_It(Value_To_Check : in POSITIVE);
        end CHECKER;

        type Checker_Ptr is access CHECKER;
        for CHECKER'STORAGE_SIZE use 256;
        Front : Checker_Ptr;

        procedure MAKE_NEW_CHECKER(A_Prime_Number : in POSITIVE;
                                   New_Checker : out Checker_Ptr);

        task body FEEDER is
                Upper_Limit : 3..INTEGER'LAST;
                Front : Checker_Ptr;
        begin
                PUT("Upper limit for primes ? ");
                GET(Upper_Limit);
                MAKE_NEW_CHECKER(2,Front);
                for COUNTER in 3..Upper_Limit loop
                        Front.Check_It(COUNTER);
                end loop;
                exception
                        when CONSTRAINT_ERROR =>
                        PUT_LINE("Erronous input; must be greater than 2");
                        when STORAGE_ERROR => PUT_LINE("Storage too small");
        end FEEDER;

        task body CHECKER is separate;

        procedure MAKE_NEW_CHECKER(A_Prime_Number : in POSITIVE;
                                   New_Checker : out Checker_Ptr) is
                RESULT : Checker_Ptr;
        begin
                RESULT := new CHECKER;
                RESULT.Who_Am_I(A_Prime_Number);
                New_Checker := RESULT;
                PUT_LINE(INTEGER'IMAGE(A_Prime_Number));
        exception
                when STORAGE_ERROR =>
                        PUT_LINE("Not enough room for new task");
        end MAKE_NEW_CHECKER;

begin
        null;
end G_PRIMES;
```

Wydruk 1

Następnie, należy zdefiniować wzorzec do tworzenia odpowiadających liczbom pierwszym zadań, które przyjmują kolejne liczby naturalne i sprawdzają ich podzielność względem tych liczb pierwszych. Jest to typ zadaniowy o nazwie *CHECKER*, którego specyfikacja zawiera dwa wejścia: wejście identyfikacyjne *Who_Am_I* i wejście przekazujące liczbę do sprawdzenia *Value_to_Check*. W ciele każdego utworzonego zadania typu *CHECKER*, które zdefiniowano jako rozłączne (wydruk 2), następuje sprawdzenie podzielności badanej liczby naturalnej i – jeśli jest niepodzielna przez daną liczbę pierwszą – przekazanie jej do kolejnego zadania typu *CHECKER* w potoku lub utworzenie odpowiadającego jej nowego zadania w potoku, o ile aktualne zadanie jest ostatnie w ciągu (ponieważ w tym wypadku badana liczba jest liczbą pierwszą).

```
separate (G_PRIMES)
task body CHECKER is

        My_Prime,Value_to_Check : POSITIVE;
        Next_Checker : Checker_Ptr;
        Prime : BOOLEAN;

begin
        accept Who_Am_I(My_Prime : in POSITIVE) do
                CHECKER.My_Prime := My_Prime;
        end Who_Am_I;

        loop

        select
                accept Check_It(Value_to_Check : in POSITIVE) do
                        CHECKER.Value_to_Check := Value_to_Check;
                end Check_It;
        or
                terminate;
        end select;

        Prime := Value_to_Check mod My_Prime /= 0;
        if Prime then
                if Next_Checker /= null
                then Next_Checker.Check_It(Value_to_Check);
                else MAKE_NEW_CHECKER(Value_to_Check,Next_Checker);
                end if;
        end if;

        end loop;

exception
        when others => PUT_LINE("Error in Checker");
end CHECKER;
```

Wydruk 2

Ostatnim krokiem jest zdefiniowanie procedury do tworzenia nowych zadań (*MAKE_NEW_CHECKER*). Oczywiście należy pamiętać, że tworzenie zadań może odbywać się tylko przez wskazania (por. typ *Checker_Ptr*).

W specyfikacji procedury głównej przedstawionej na wydruku 1 interesujące jest jeszcze użycie atrybutu *STORAGE_SIZE* w celu zmniejszenia wielkości stosu dla poszczególnych zadań, oraz – atrybutu *IMAGE* do wyprowadzenia wartości liczby. W pamięci o pojemności 640 KB komputera IBM PC mieści się 75 zadań utworzonych w ten sposób.

Oprac. JANUSZ ZALEWSKI

Simon H. A.: Procesy przeszukiwania i wnioskowania w rozwiązywaniu problemów (1)

INFORMATYKA 1989, nr 3, s.1

Pierwsza część artykułu z dziedziny badań nad sztuczną inteligencją na temat zależności między procesami poszukiwania rozwiązań i wnioskowania w rozwiązywaniu problemów.

Rybnik J., Solak J.: Stacje robocze Sun-3 (1). Architektura

INFORMATYKA 1989, nr 3, s.4

Pierwsza część artykułu na temat charakterystycznych cech stacji roboczych, zawierająca omówienie architektury modeli rodziny stacji Sun-3.

Głębicki J.: System dBase III Plus w sieciach lokalnych

INFORMATYKA 1989, nr 3, s.8

Charakterystyka rozwiązań oraz sposobu działania systemu bazy danych w sieciach lokalnych.

Gizdoń H., Pawlak A., Wrona W.: Zastosowanie języka VHDL do opisu i weryfikacji projektów układów cyfrowych

INFORMATYKA 1989, nr 3, s.14

Praktyka stosowania języka VHDL na przykładzie wybranego układu cyfrowego o dużym stopniu scalenia.

Nowiński W.: Rekonstruowanie obrazów w tomografii komputerowej

INFORMATYKA 1989, nr 3, s.18

Charakterystyka tomografii komputerowej oraz stosowanych rozwiązań systemu rekonstruowania obrazów.

Bielecki J.: Turbo C. Jawne i niejawne konwersje danych

INFORMATYKA 1989, nr 3, s.21

Kontynuacja praktycznych wskazówek i przykładów stosowania niektórych specyficznych rozwiązań języka programowania Turbo C.

Sikorski A.: Lokalne sieci światłowodowe

INFORMATYKA 1989, nr 3, s.24

Historia i perspektywy rozwoju lokalnych sieci światłowodowych oraz charakterystyka najbardziej znanych rozwiązań takich sieci.

Simon H. A.: Search and reasoning in problem solving (1)

INFORMATYKA 1989, No. 3, p.1

First part of the paper from the field of AI research on interdependence between solution search and reasoning processes in problem solving.

Rybnik J., Solak J.: Sun-3 workstations (1). The architecture

INFORMATYKA 1989, No. 3, p.4

First part of the paper on characteristic features of workstations, which includes discussion of the architecture of the Sun-3 workstation family models.

Głębicki J.: dBase III Plus system in local area networks

INFORMATYKA 1989, No. 3, p.8

Characteristics of data base system solutions and operation methods in local area networks.

Gizdoń H., Pawlak A., Wrona W.: Application of the VHDL language for description and verification of digital circuit design

INFORMATYKA 1989, No. 3, p.14

The VHDL language application practice on example of chosen LSI circuit.

Nowiński W.: Image reconstructing in tomography

INFORMATYKA 1989, No. 3, p.18

Characteristics of computer tomography and applied solutions of image reconstruction system.

Bielecki J.: Turbo C. Open and closed data conversion

INFORMATYKA 1989, No. 3, p.21

Continuation of practical advices and examples for application of some specific solutions of the Turbo C language.

Sikorski A.: Fiber-optic local area networks

INFORMATYKA 1989, No. 3, p.24

History and perspectives of the fiber-optic local area networks and characteristics of the most known solutions of such networks.

Simon H. A.: Suchungs- und Schlussfolgerungsprozesse bei Lösung von Problemen (1)

INFORMATYKA 1989, Nr. 3, S.1

Erster Teil eines Artikels aus dem Bereich der künstlichen Intelligenz über Abhängigkeit zwischen Prozessen der Lösungensuchung und Schlussfolgerung bei Lösung von Problemen.

Rybnik J., Solak J.: Sun-3-Arbeitsplatzcomputer (1). Die Architektur

INFORMATYKA 1989, Nr. 3, S.4

Erster Teil eines Artikels über charakteristische Eigenschaften der Arbeitsplatzcomputer, der eine Besprechung der Architektur von Modellen der Sun-3-Computerfamilie umfasst.

Głębicki J.: dBase III Plus-System in lokalen Netzen

INFORMATYKA 1989, Nr. 3, S.8

Eine Charakteristik von Lösungen und Betriebsmethode des Datenbanksystems in lokalen Netzen.

Gizdoń H., Pawlak A., Wrona W.: Anwendung von VHDL-Sprache für Beschreibung und Verifikation der Digitalschaltkreiseprojektierung

INFORMATYKA 1989, Nr. 3, S.14

Anwendungspraxis der VHDL-Sprache an Beispiel einer ausgewählten LSI-Schaltung.

Nowiński W.: Wiederherstellung von Bildern in Computertomografie

INFORMATYKA 1989, Nr. 3, S.18

Eine Charakteristik der Computertomografie und der dort angewendeten Lösungen des Bildwiederherstellungssystems.

Bielecki J.: Turbo C. Offene und nicht offene Datenumwandlung

INFORMATYKA 1989, Nr. 3, S.21

Fortsetzung von praktischen Hinweisen und Beispielen über spezifische Lösungen der Turbo C-Sprache.

Sikorski A.: Lokale Lichtwellenleiternetze

INFORMATYKA 1989, Nr. 3, S.24

Geschichte und Aussichten der Entwicklung von lokalen Lichtwellenleiternetzen und eine Charakteristik der am meisten bekannten Lösungen solcher Netze.

Słowniki, które warto wziąć do ręki

W bieżącym wydaniu rubryki terminologicznej polecam Czytelnikom trzy słowniki, do których warto zajrzeć, jeśli znajdą się w zasięgu ręki, a może nadarzyć się taka okazja. Choć w ogóle przestrzegałbym przed bezkrytycznym używaniem zagranicznych, także wymienionych poniżej, słowników terminologicznych (o ile nie są jednocześnie normami terminologicznymi), ponieważ są bardzo nieścisłe – odgrywają one nieocenioną rolę jako poradniki ze względu na zawarte w nich zasoby wiedzy, które należy skonsultować przed podjęciem decyzji, jak nazwać odpowiednie pojęcie lub w jakim właściwym znaczeniu używać odpowiedniego terminu. Podkreślam jednak, że w pełni autorytatywnym źródłem mogą być tylko normy terminologiczne (np. ISO 2382, IEEE 610), co nie umniejsza wszakże roli słowników pod wymienionym względem, choćby dlatego, iż normy powstają z pewnym opóźnieniem i mogą się dezaktualizować. **(J. Zal.)**

F. J. Galland (Ed.), Dictionary of Computing – Data Communications, Hardware and Software Basics, Digital Electronics. Wiley and Sons, Chichester (UK), 1982. ISBN 0471-10469-8

Słownik zawiera ponad 9000 haseł wraz z definicjami i objaśnieniami odpowiednich pojęć z techniki obliczeniowej, obejmującej elektronikę cyfrową, podstawy sprzętu i oprogramowania oraz teleinformatykę. Jego celem jest ułatwienie czytelnikom korzystania z książek, czasopism, a szczególnie z podręczników firmowych i opisów technicznych. Jako taki powinien służyć głównie studentom informatyki, elektroniki, telekomunikacji i dziedzin pokrewnych, lecz może bardzo przydać się również osobom zatrudnionym przy obsłudze sprzętu komputerowego, biznesmenom, menedżerom, a także autorom książek z wymienionych dziedzin. Autorami haseł są specjaliści z różnych gałęzi informatyki, jak oprogramowanie dużych komputerów i minikomputerów, systemy mikroprocesorowe, sprzęt i oprogramowanie komunikacyjne, przetwarzanie tekstów itp., mający ponadto dużo doświadczenia w pisaniu podręczników i w szkoleniu użytkowników, co zapewnia słownikowi poziom rzetelnej wiedzy informatycznej – i jak powiedział jeden z czytelników – odsłania tajemnice żargonu rodem z pomieszczeń komputerowych. Dodając do tego dużą staranność znanego wydawnictwa Wiley and Sons w przygotowaniu edycji, otrzymuje się bardzo wygodną pomoc w nauce i w pracy.

J.M. Rosenberg, Dictionary of Computers, Information Processing and Telecommunications. Second Edition. Wiley and Sons, Chichester (UK), 1987. ISBN 0471-85558-8

Ten słownik uważa się za jeden z nielicznych, które obejmują trzy gałęzie składające się w rozumieniu autora na technologię informacyjną: technikę komputerową, przetwarzanie informacji i telekomunikację. Nowe wydanie zawiera 12000 terminów (w tej liczbie 2000 nowych) wraz z definicjami i objaśnieniami pojęć, między innymi z takich wyłaniających się dopiero dziedzin, jak lokalne sieci komputerowe, cyfrowe przekazywanie informacji głosowej, sztuczna inteligencja i systemy ekspertowe, superkomputery i in. Opracowany przy udziale i przy pomocy różnych agend rządowych, odpowiedzialnych za rozwój technologii informacyjnej, słownik odzwierciedla takie znaczenia terminów, w jakich są używane w bieżącej praktyce. Istotną zaletą słownika jest rozbicie objaśnień do wielu haseł na dwie definicje: ogólną, wyjaśniającą laikowi znaczenie terminu na poziomie nietechnicznym i szczegółową, dostarczającą precyzyjnej informacji ekspertom.

S. Ishibashi (Ed.), Dictionary of Electronics and Electrical Engineering – English – Japanese – German – Russian. Third Edition. Plenum Press, New York, 1987. ISBN 0-306-42749-4

Trzecie wydanie tego słownika zawiera ponad 42 000 haseł używanych obecnie w elektronice, elektrotechnice i dziedzinach pokrewnych, jak telekomunikacja, fizyka, automatyka, informatyka, nukleonika i in. Hasła są uporządkowane alfabetycznie według ich brzmienia w języku angielskim i zawierają odpowiedniki japońskie, niemieckie i rosyjskie, jednak oddzielne indeksy umożliwiają dla każdego z tych trzech języków łatwe wyszukiwanie odpowiedników angielskich (według numeru strony i wiersza). Dodatkowo, słownik zawiera tabelę powszechnie używanych akronimów i skrótów, wraz z wyjaśnieniem ich znaczenia i tłumaczeniem na wymienione języki. Są w nim także tabele jednostek miar oraz liczb we wszystkich czterech językach. Poleca się go szczególnie inżynierom, pracownikom naukowym i tłumaczom, a także studentom, menedżerom i innym osobom związanym zawodowo z wymienionymi dziedzinami.

D. Longley, M. Shain, Data and Computer Security. Dictionary of Standards, Concepts and Terms. Macmillan, New York, 1988. ISBN 0-333-42935-4

Wraz z rozwojem zastosowań komputerów i sieci łączności danych, w ostatnich latach wzrosło zainteresowanie ochroną i zabezpieczaniem informacji komputerowych. Choć pierwszym koniecznym krokiem w tym kierunku było i jest rozwiązywanie problemów technicznych, to prowadzi ono nieuchronnie do wykształcenia personelu rozumiejącego i stosującego te techniki i procedury. Niniejszy słownik, stanowiący niemal kompletny zbiór terminów i pojęć, służy właśnie profesjonalistom związanym z zabezpieczaniem danych i komputerów. Specjaliści zajmujący się tą dziedziną znajdą w nim wyjaśnienie pojęć matematycznych dotyczących nowoczesnego zabezpieczania danych, np. z teorii liczb i teorii złożoności, i omówienie ich zastosowania w nowych opracowaniach z technologii informacyjnej. Jednocześnie, niespecjaliści znajdą w słowniku szczegółowe wyjaśnienie terminów, wyrażeń żargonowych i akronimów oraz wiele praktycznych rad z tej dziedziny.

Oprócz typowych objaśnień poszczególnych haseł, słownik zawiera szereg oddzielnych artykułów omawiających najważniejsze pojęcia zabezpieczania danych, jak utajnianie danych, autoryzowanie danych, stosowanie kluczy, i pogłębiających wiedzę w takich dziedzinach, jak analiza języka, metodyka macierzowa, zabezpieczenie baz danych, zabezpieczanie sieci i sterowanie dostępem. Ponadto słownik obejmuje również pojęcia dotyczące zabezpieczania łączności w bankowości, ubezpieczeniach i przetwarzaniu danych. Autorzy omawiają także szczegółowo aktualne normy ANSI dotyczące zabezpieczania danych oraz terminologię stosowaną w Departamencie Obrony USA, lecz użyteczną także w zastosowaniach cywilnych.

Słownik zawiera alfabetyczny wykaz 3500 haseł z dziedziny zabezpieczania danych i komputerów. Szerokie zastosowanie odsyłaczy do haseł pokrewnych zapewnia łatwy dostęp do dowolnej informacji. Szczegółowy indeks i kilkadziesiąt rysunków stanowi istotne uzupełnienie słownika, dzięki czemu Czytelnik może utworzyć sobie szybko pełny obraz dowolnego fragmentu tej dziedziny.

Istitute of Electrical and Electronics Engineers, Standard Dictionary of Electrical and Electronics Terms. Fourth Edition. IEEE, New York, 1988. ISBN 0471-85558-8

Czwarte wydanie tego słownika obejmuje definicje prawie 25 000 terminów z dziedziny elektryki, elektroniki i techniki komputerowej. Hasła są ułożone alfabetycznie i zawierają oprócz definicji wiele użytecznych informacji, dotyczących np. zalecanego użycia, różnic znaczeniowych w poszczególnych dziedzinach, a także – odsyłacze do odpowiednich prac źródłowych i określenie pochodzenia terminów. W wielu definicjach dodano też wyjaśnienia omawiające bardziej szczegółowo problemy stosowania odnośnej terminologii. Oprócz zwykłych haseł słownik zawiera ponad 15 000 akronimów, używanych w nauce, przemyśle, administracji i wojsku. Podane definicje są oficjalnie przyjęte przez Amerykański Instytut Normalizacyjny (słownik jest normą ANSI/IEEE Std 100-1988), jednak w wielu wypadkach zostały przeniesione wprost z zaleceń Międzynarodowej Komisji Elektrotechnicznej. Słownik jest przeznaczony dla inżynierów i techników elektroników, lecz może stanowić także nieocenioną pomoc dla nauczycieli i studentów specjalności elektronicznych i elektrycznych.

Transoceaniczne kable światłowodowe

Do 1956 r. można było porozumiewać się między kontynentami jedynie drogą radiową. Pierwszy kabel transatlantycki o nazwie TAZ-1, który kosztował prawie 50 mln dolarów i łączył Szkocję z Nową Funlandią, umożliwił przenoszenie tylko 52 rozmów. Później położono wiele innych kabli, z których ostatni zwany TAT-7 został zbudowany w 1983 r. za ponad 190 mln dolarów i przenosił już 90 tysięcy rozmów. Wszystkie te kable nie zaspokajają jednak potrzeb i obecnie około 60% połączeń dokonuje się przez satelity komunikacyjne, które wprowadzają około półsekundowe opóźnienie (odległość satelity prawie 36 tys. km) i zjawisko echa.

Rozwiązaniem tych problemów wydaje się być kładziony obecnie kabel światłowodowy, który miał być oddany do użytku w 1988 r., a który przy kosztach rzędu 335 mln dolarów ma przenosić 40 tysięcy rozmów telefonicznych lub równoważne ilości przesyłanej informacji.

Kabel ten będzie własnością 29 przedsiębiorstw amerykańskich i europejskich, wśród których największy udział ma firma AT&T (37%). Buduje ona największy odcinek kabla długości ok. 5860 km, z Tuckerton w stanie New Jersey do punktu u wybrzeży Europy, w którym kabel rozdziela się na odcinki prowadzące do Wielkiej Brytanii i Francji, budowane przez firmy łącznościowe z tych krajów.

W USA zainstalowano już ponad 37 tysięcy kilometrów kabli światłowodowych łączących główne miasta tego kraju. Zwiększenie możliwości przesyłowych dzięki zastosowaniu kabla światłowodowego obniży koszt przesyłania danych i koszt rozmów telefonicznych (aktualnie 8 dolarów za 10 minut z Manhattanu do Londynu), jak również umożliwi przesyłanie sygnałów wizyjnych. Ponadto zapewni to ochronę przesyłanej informacji przed niepowołanym dostępem, co jest istotne np. dla banków. Informacja nie może być przechwycona jak w przypadku transmisji satelitarnej, a z samego światłowodu nie można jej łatwo uzyskać.

Jako wady łączności światłowodowej wymienia się możliwość uszkodzenia, co wymagałoby długotrwałej lokalizacji i naprawy. Ciekawostką jest, iż podczas badań koło wysp Kanaryjskich stwierdzono jakieś oddziaływanie kabla na przepływające w pobliżu rekiny, które próbowały go atakować. Z tego względu zdecydowano się na solidniejsze zabezpieczenie zewnętrzne niż początkowo planowano. Najbardziej narażony na uszkodzenia będzie odcinek ok. 160 km od brzegu USA zalegający płytko na szelfie kontynentalnym. Dlatego zdecydowano się zakopać tu kabel na głębokości ok. 0,6 m za pomocą specjalnego robota-pługa. Po zakończeniu tego odcinka, układanego z kanadyjskiego statku, do działania przystąpił statek kablowy firm AT&T o nazwie Lorg Lines; kabel jest układany na dnie oceanu z szybkością prawie 10 km/godz. Trasę wytyczono tak, by uniknąć gwałtownych zmian głębokości i pasm wulkanicznych.

Nieco później rozpoczęte prace nad światłowodowym kablem przez Ocean Spokojny, o łącznej długości ok. 16 tysięcy kilometrów, który ma być ukończony w 1989 r. Będzie on łączyć Point Arena w Kaliforni z Hawajami (4200 km), a następnie zostanie prowadzony ok. 5400 km na zachód do punktu rozgałęzienia w kierunku Japonii (ok. 2200 km) i do Filipin (4200 km) przez wyspę Guam (1600 km). Koszt przedsięwzięcia ma wynosić 700 mln dolarów.

Po ukończeniu obu tych przedsięwzięć będzie możliwe korzystanie z zasobów informacji w dowolnym zakątku kuli ziemskiej.

J.R.

Szybka i niezawodna drukarka diodowa

Większość drukarek bezuderzeniowych jest oparta na technice laserowej; natomiast firma Eastman Kodak zbudowała swą najnowszą bezuderzeniową drukarkę elektroniczną z zastosowaniem techniki diod świecących (LED – *Light Emitting Devices*). Urządzenie to o nazwie Ektaprint 1392 jest przeznaczone do szybkiego drukowania dużych ilości informacji. Szybkość drukowania wynosi 92 strony na minutę z rozdzielczością 300 elementów na cal (czyli 12 na mm).

Według R. Vukosica z Działu Urządzeń Kopiujących przy opracowaniu tej drukarki było znacznie mniej problemów inżynierskich niż przy jej odpowiednikach laserowych. Zamiast bowiem trudnych do wykonania i drogich podzespołów, jak laser, modulator, soczewki i skaner, omawiana drukarka ma tzw. wiersz świetlny, który naświetla powierzchnię fotoprzewodzącą wzdłuż całej szerokości. Ponadto w drukarkach laserowych czułym punktem jest wielokątny skaner, którego powierzchnie muszą być precyzyjnie obrobione, a płaszczyzna obrotu musi być utrzymana z dużą dokładnością. Natomiast drukarka diodowa nie ma części ruchomych poza mechanizmem przesuwania papieru.

Działanie drukarki polega na szeregowym przesyłaniu pełnego wiersza danych do głowicy drukującej; następnie odpowiednie układy przesyłają te dane w postaci równoległej do rejestrów przesuwnych, gdzie informacja jest przechowywana aż do otrzymania sygnału wyzwalającego naświetlanie. Wówczas zapalają się odpowiednie diody zgodnie z zakodowanym obrazem. Przy szybkiej reakcji diod – rzędu 5μs – system daje lepszą dokładność niż odpowiedniki laserowe. Każda dioda świeci się przez bardzo krótki okres – np. 0,026 s na stronę o wymiarach 21,6×28 cm. Przyjmując średni okres żywotności diody równy 1000 godzin uzyskuje się możliwość druku ponad 138 milionów stron.

J.R

Prognoza rozwoju pamięci na dyskach

Zajmująca się analizą rynku komputerowego firma Freeman Associates Ins. z Santa Barbara w Kalifornii opracowała studium „Optical Data Storage Outlook", w którym podane są przewidywania odnośnie ilościowego i wartościowego rozwoju dyskowych pamięci optycznych w najbliższych latach.

Rozróżnia się trzy rodzaje pamięci na dyskach optycznych: stałą, z jednokrotnym zapisem i z wielokrotnym zapisem. Przewiduje się, że w latach 1985–1991 średni wzrost sprzedaży stacji dla wszystkich tych rodzajów dysków wyniesie 105% rocznie. W 1991 r. najwięcej będzie stacji dysków stałych (442 tysiące, tj. 42%), następnie stacji dysków z jednokrotnym zapisem (345 tysięcy, 33%) i stacji dysków z wielokrotnym zapisem (260 tysięcy, 25%). Urządzenia pamięci stałej mają kosztować 174 mln dolarów i stanowią tylko 9% wartości wszystkich stacji. Natomiast stacje dysków z jednokrotnym zapisem odpowiadają wartości 1,5 mld dolarów, co stanowi 74% całości. Wpływy za urządzenia z wielokrotnym zapisem mają wynieść w 1991 r. 347 mln dolarów, tj. 17%. Tak więc, udział systemów z zapisem wzrośnie, jeśli chodzi o wartość, z 71% w 1985 do 91% w 1991 r. Wśród stacji dysków stałych będą dominować (95%) urządzenia kompaktowe (CD). Wśród producentów dominują tu firmy japońskie oraz Philips, które mają ustaloną pozycję na rynku fonograficznym odnośnie sprzętu tego rodzaju. Niektóre firmy amerykańskie podejmą działalność w tej dziedzinie, ale będzie to dotyczyć wydruku elektronicznego, produkcji samych dysków i rozbudowanych systemów, a nie stacji.

Wartość sprzedanych w 1985 r. stacji dysków z jednokrotnym zapisem dwukrotnie przekraczała wartość stacji dysków stałych. W 1986 r. stosunek ten miał wynosić 4, a w 1991 r. – już 8.

Dyski z jednokrotnym zapisem dzielą się z kolei na trzy grupy w zależności od pojemności, przy czym wartościami granicznymi są 1 i 3 GB. Najwcześniej rozwinięto produkcję pamięci, których pojemność zawiera się między 1 a 3 GB. Te właśnie pamięci stanowią wartościowo największą pozycję w całym okresie objętym prognozą. W 1987 r. sprzedano więcej stacji dysków o pojemności mniejszej niż 1 GB, jednak ze względu na znaczną obniżkę ceny, stacje te stanowić będą w 1991 r. tylko 17% wpływów.

Urządzenia o pojemności ponad 3 GB pojawią się w niewielkich ilościach pod koniec lat osiemdziesiątych dla specjalnych zastosowań, a dyskowe pamięci optyczne z wielokrotnym zapisem pojawiły się na rynku poczynając od 1987 r. W większości są to systemy o pojemności mniejszej niż 1 GB. W 1990 r. pojawią się pierwsze urządzenia tego typu o większych pojemnościach, jako specjalizowane moduły dużych komputerów.

Wartość wyprodukowanych pamięci z wielokrotnym zapisem przekroczy poziom pamięci stałych na dyskach optycznych w 1990 r. a w rok później suma uzyskana ich sprzedaży będzie dwukrotnie większa niż wartość pamięci stałych.

J.R.