STUDIA INFORMATICA 2005
Volume 26 Number 2 (63)

Dariusz STRZĘCIWILK
Szkoła Główna Gospodarstwa Wiejskiego, Katedra Ekonometrii i Informatyki

KOMPUTEROWE WSPOMAGANIE NAUCZANIA Z ZAKRESU KOMUTATORÓW

Streszczenie. W artykule przedstawiono przykład rozwiązania systemu wspomagającego nauczanie z zakresu komutatorów. Zaprezentowano możliwości systemu dydaktycznego umożliwiającego wszechstronne ćwiczenia w ramach tematyki multiplekserów i demultiplekserów. Omówiono również możliwości systemu wspomagającego nauczanie z zakresu zagadnień komutatorów do realizacji funkcji logicznych.

Słowa kluczowe: systemy wspomagające nauczanie, komutatory, multipleksery, demultipleksery.

COMPUTER AIDED TEACHING OF COMMUTATORS

Summary. The article describes an example of a solution for system aided teaching of commutators. Possibilities and versatility of exercises of didactic program used for teaching of multiplexers and demultiplexers have been presented. Possibilities of didactic system for teaching of commutators in realizations of logical functions were presented too.

Keywords: system aided teaching, commutators, multiplexers, demultiplexers.

1. Wprowadzenie

Szybki postęp techniki cyfrowej sprawia, iż układy scalone średniej skali integracji – MSI (*medium scale integration*) oraz dużej skali integracji – LSI (*large scale integration*) znajdują szerokie zastosowanie w realizacji złożonych automatów cyfrowych [1]. Mogą być użyte również do realizacji funkcji logicznych spotykanych w układach sterowania i to zarówno w układach kombinacyjnych, jak i sekwencyjnych. Do tego celu można wykorzystać następujące elementy scalone:

- multipleksery i demultipleksery,
- pamięci półprzewodnikowe,
- programowane pola logiczne.

Wymienione tu układy znajdują rozmaite zastosowania w technice cyfrowej. Multipleksery bywają wykorzystywane np. w układach do przesyłania informacji, pochodzących z wielu źródeł, za pomocą linii jednoprzewodowej. Demultipleksery służą do przesyłania informacji z jednego źródła do wielu odbiorników. Ponadto demultiplekser, stający się w łatwy sposób dekoderem kodu binarnego na kod 1 z *n*, znajduje zastosowanie w układach zmieniających kod binarny na kody inne niż 1 z *n*. Coraz większe znaczenie tych układów spowodowało zainteresowanie nimi nie tylko ze względu na zastosowanie w maszynach cyfrowych, ale również do realizacji układów sterowania i innych [2, 3]. Istnieje więc potrzeba szerokiego rozpowszechnienia metod projektowania układów cyfrowych oraz realizacji funkcji logicznych za pomocą komutatorów (multiplekserów i demultiplekserów). W procesach nauczania i uczenia się często wykorzystuje się komputerowe wspomaganie nauczania [4, 5]. Wśród programów i systemów nauczania wspomaganego komputerowo i uczenia się wspomaganego komputerowo wyróżniamy systemy CAI (*computer assisted instruction*) i CAL (*computer assisted learning*). Systemy te pozwalają na dostosowanie szybkości uczenia się do możliwości i zdolności ucznia. Programy edukacyjne możemy podzielić na trzy grupy [6]. Do pierwszej grupy możemy zaliczyć programy demonstracyjne i symulatory. Stosowane są one w celu przedstawienia zjawisk lub procesów niemożliwych do przedstawienia w warunkach naturalnych. W wielu przypadkach przeprowadzenie eksperymentu na układzie jest niemożliwe lub niepraktyczne z uwagi na niebezpieczeństwo wykonania eksperymentu, lub koszt eksperymentu. Programy z tej grupy obrazują przebieg zjawisk i realizują je w sposób dokładny lub przybliżony, systemem opisanym pewnym modelem teoretycznym. Wymagają one od użytkownika pewnego przygotowania teoretycznego oraz umiejętności często złożonej obsługi. Do drugiej grupy zaliczmy programy ukierunkowane na ćwiczenia i doskonalenie umiejętności. Stosowane są one w celu pamięciowego opanowania pewnych pojęć, czynności lub operacji. Komunikacja użytkownika z programem odbywa się za pomocą wprowadzania danych, a informacje o stanie wiedzy i zdobytych umiejętnościach użytkownika wyprowadzane są w postaci komunikatów. Do ostatniej grupy zaliczmy programy typu korepetytor. W programach tych użytkownik zapoznaje się z określonym materiałem, podzielonym na odpowiednio małe fragmenty. Po zapoznaniu z nim użytkownik sprawdza stopień opanowania materiału za pomocą pytań sprawdzających i ćwiczeń praktycznych. Zaletą tego typu programów jest duża łatwość obsługi oraz możliwość indywidualizacji procesu nauczania.

W prezentowanym systemie podjęto próbę połączenia wszystkich cech opisanej grupy programów. Pozwala on użytkownikowi na samodzielne tworzenie plików z zadaniami

o różnym stopniu trudności oraz na samodzielne rozwiązywanie zadań. W części demonstracyjnej użytkownik może prześledzić zadania rozwiązywane w sposób automatyczny. Teoretyczne przedstawienie zagadnień związanych z komutatorami dostępne jest w części teoretycznej programu.

2. Koncepcja realizacji programu

W opracowanym programie przyjęto założenia, iż użytkownik będzie decydował o rodzaju multipleksera i demultipleksera (4, 8, lub 16 bitów) oraz liczbie zmiennych dla realizacji danej struktury. Ze względu na rozmiar produkowanych układów, w oparciu o pojedynczy multiplekser lub demultiplekser, można zrealizować funkcje logiczne o liczbie argumentów $p \le 4$. Z tego względu zostały uwzględnione następujące struktury układów:

- pojedynczy multiplekser,
- pojedynczy demultiplekser.

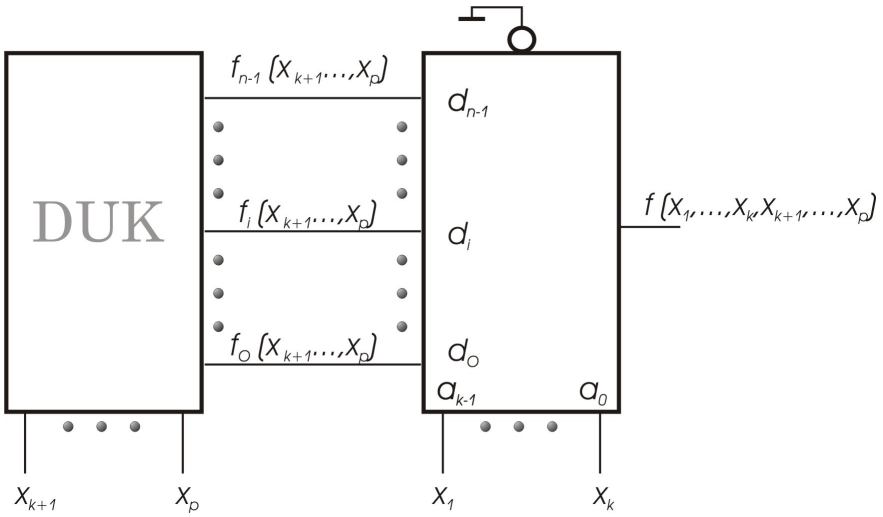
W przypadku gdy liczba argumentów funkcji $p > 4$, należy zastosować dodatkowy układ kombinacyjny (DUK). W takiej sytuacji DUK może być realizowany z zastosowaniem bramek, demultipleksera i bramek lub multiplekserów. Aby zrealizować funkcję logiczną $f(x_1, ..., x_k, x_{k+1}..., x_p)$ o liczbie argumentów $p > 4$ opierając się na multiplekserach o k wejściach adresowych, należy funkcję rozłożyć względem tych k argumentów, które będą wprowadzone na wejścia adresowe multipleksera:

$$f(x_1,...,x_k,x_{k+1},...,x_p) = \overline{x_1},...,\overline{x_k} f(0,...,0,x_{k+1},...,x_p) + \overline{x_1},...,x_k f(0,...,1,x_{k+1},...,x_p) + ...$$
$$+ x_1...x_k f(1,...,1,x_{k+1},...,x_p) = S_0(x_1,...,x_k) f_0(x_{k+1},...,x_p) + S_1(x_1,...,x_k) f_1(x_{k+1},...,x_p) + ...$$
$$+ S_{n-1}(x_1,...,x_k) f_{n-1}(x_{k+1},...,x_p) = \sum_{i=0}^{n-1} S_i(x_1,...,x_k) f_i(x_{k+1},...,x_p)$$

oraz przyjąć

$$d_i = f_i(x_{k+1},...,x_p).$$

Realizację tak rozłożonej funkcji przedstawiono schematycznie na rysunku 1. Dodatkowy układ kombinacyjny generuje funkcje $f_i(x_{k+1}, ..., x_p)$, a na wejścia adresowe multipleksera wprowadzane są argumenty $x_1, ..., x_k$, zgodnie z ich wagami.



Rys. 1. Schemat blokowy struktury z DUK
Fig. 1. Block diagram of DUK structure

W opracowanym programie przyjęto, że DUK będzie realizowany z zastosowaniem układów:

- multiplekser i demultiplekser,
- układ bramek i demultiplekser.

DUK można więc realizować w strukturach „Układ bramek i multiplekser" oraz „Multiplekser i Demultiplekser". W strukturach tych można realizować funkcje o maksymalnej liczbie argumentów ≤ 8. Podfunkcje $f_i(x_{k+1}, ..., x_p)$ generowane są przez jeden demultiplekser lub przez układ bramek, którym można przyporządkować maksymalnie cztery argumenty. Do realizacji funkcji w tych strukturach wymagane jest przeprowadzenie minimalizacji wprowadzonej funkcji logicznej. W tym celu w programie, zastosowano algorytm minimalizacji metodą bezpośredniego przeszukiwania. Określenie rozwiązania dla DUK zaimplementowano w oparciu o następujący algorytm [7, 8].

Algorytm realizacji funkcji F w oparciu o dodatkowy układ kombinacyjny

1. Wybrać argumenty podstawowe i przyporządkować je wejściom adresowym multipleksera. Aby uzyskać optymalne rozwiązanie DUK, za argumenty podstawowe należy wybrać te, które w minimalnej postaci funkcji występują najczęściej. Wejściu strobującemu $\overline{S}$ przyporządkować wartość 0.
2. Wpisać wartości funkcji do siatki Karnaugha zorganizowanej w taki sposób, że argumenty podstawowe kodują kolumny (wiersze), a pozostałe argumenty zwane dodatkowymi kodują wiersze (kolumny).
3. Ponumerować kolumny i wiersze zgodnie z wagami wynikającymi z punktu 1.
4. W oparciu o *i*-tą kolumnę (wiersz) należy określić funkcję dla *i*-tego wejścia informacyjnego multipleksera:

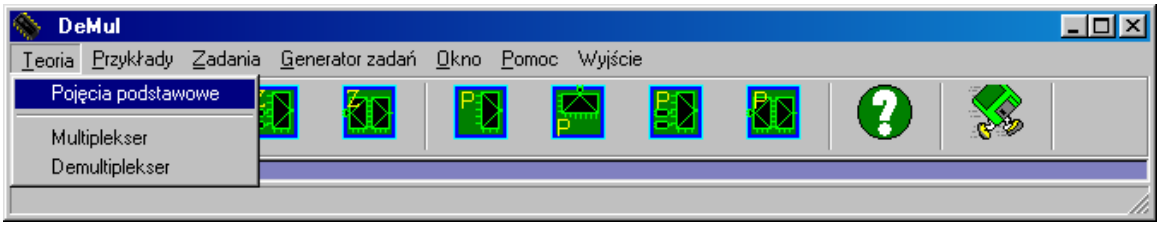
– W postaci minimalnego wyrażenia, gdy ma ona być realizowana w oparciu o układ bramek.
– W postaci kanonicznej i zastosować odpowiedni algorytm realizacji funkcji F w postaci iloczynu lub sumy [7, 8], gdy ma być realizowana w oparciu o multiplekser lub demultiplekser.

3. Opis programu

Program DeMul (dostępny w witrynie internetowej: http://zmitac.iinf.polsl.gliwice) napisany został jako aplikacja dla środowiska Windows 98 wspomagająca nauczanie z zakresu komutatorów. Jako podstawowe narzędzie programistyczne wybrano 32-bitowe wizualne środowisko Delphi firmy Borland. Aplikacje Win32 GUI wykorzystują wszystkie zalety kodu języka Object Pascal, który wkomponowany został w otoczenie RAD. Program DeMul jest aplikacją typu MDI (Multiple Documetn Intrface) składającą się z wielu formularzy. Pracuje poprawnie także w środowisku Windows 95, ME oraz XP. Aby wykorzystać system pomocy dla aplikacji, niezbędne są również pliki pomocy.

3.1. Interfejs użytkownika

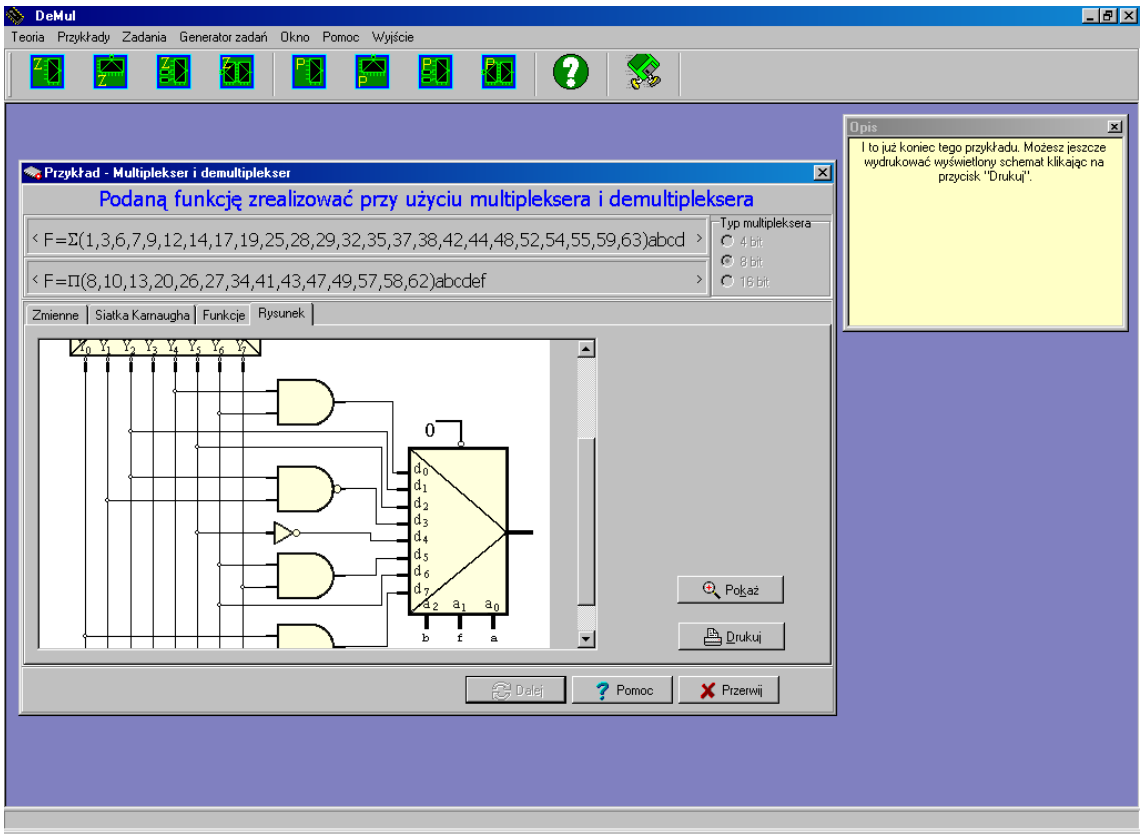
Po uruchomieniu aplikacji DeMul pojawia się okno główne programu. W górnej części okna aplikacji znajduje się pasek tytułu oraz menu główne programu. Na pasku tytułu zlokalizowane są przyciski menu sterowania dla aplikacji, pozwalające określić położenie oraz rozmiar okna programu. Poniżej paska tytułu znajduje się menu główne programu (rys. 2), które umożliwia dostęp do wszystkich funkcji programu.



Rys. 2. Okno główne programu
Fig. 2. Main window of the program

Opcja teoria pozwala na zapoznanie się z zagadnieniami teoretycznymi związanymi z komutatorami oraz objaśnia podstawowe pojęcia niezbędne do zrozumienia tematu. W opcji tej dostępne są polecenia „Pojęcia podstawowe”, „Multiplekser” oraz „Demultiplekser”, widoczne w rozwinięciu menu tej opcji na rysunku 2. Kolejna opcja pozwala na zapoznanie się z przykładami zadań dla wybranych struktur. W opcji tej dostępne są polecenia „Multiplek-

ser", „Demultiplekser", „Układ bramek i multiplekser" oraz „Multiplekser i Demultiplekser". Aby prześledzić sposób realizacji funkcji logicznej w wybranej strukturze, należy postępować zgodnie z opisem zawartym w okienku dialogowym „Opis". Kolejne etapy realizacji wszystkich przykładów przedstawiono w formie zakładek, które udostępniane są po naciśnięciu przycisku „Dalej". Wybranie opcji „Pomoc" na pierwszej zakładce przykładu umożliwia wyświetlenie postaci funkcji zminimalizowanych (postać minimalna sumy oraz postać minimalna iloczynu) realizowanej funkcji logicznej. W końcowym etapie przykładu na zakładce „Rysunek" program przedstawia schemat układu wygenerowany dla realizowanej struktury (rys. 3).



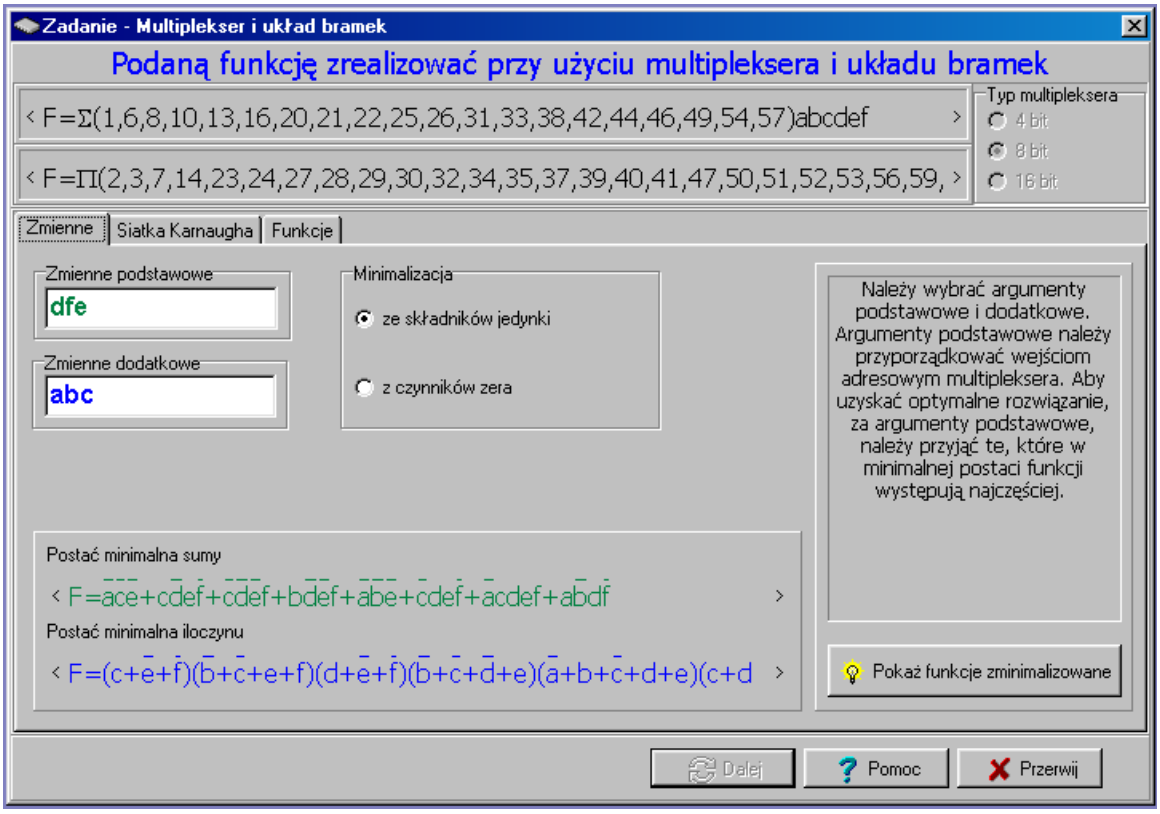
Rys. 3. Okno programu podczas prezentacji przykładu
Fig. 3. Windows shown during presentation of example

Po zakończeniu wszystkich etapów realizacji przykładu użytkownik ma możliwość wydrukowania wyników realizacji układowej wprowadzonej funkcji logicznej. Aplikacja umożliwia zapoznanie się z zagadnieniami o różnym stopniu trudności, tak aby użytkownik mógł przechodzić stopniowo od zagadnień najprostszych do bardziej złożonych.

Wybór opcji Zadania pozwala na samodzielne rozwiązywanie zadań dla wybranych struktur. Dane dla zadań są zawarte w plikach utworzonych za pomocą generatora zadań. W opcji tej dostępne są polecenia „Multiplekser", „Demultiplekser", „Układ bramek i Multiplekser" oraz „Multiplekser i Demultiplekser". Realizacja zadań w strukturze Multi-

plekser, Demultiplekser odbywa się w kilku etapach. W pierwszym etapie użytkownik musi wskazać typ multipleksera (demultipleksera) dla realizowanej struktury, a w kolejnych wskazać numery wejść podłączonych do stanu „1” i stanu „0”. W końcowym etapie generowany jest schemat układu, który można również wydrukować.

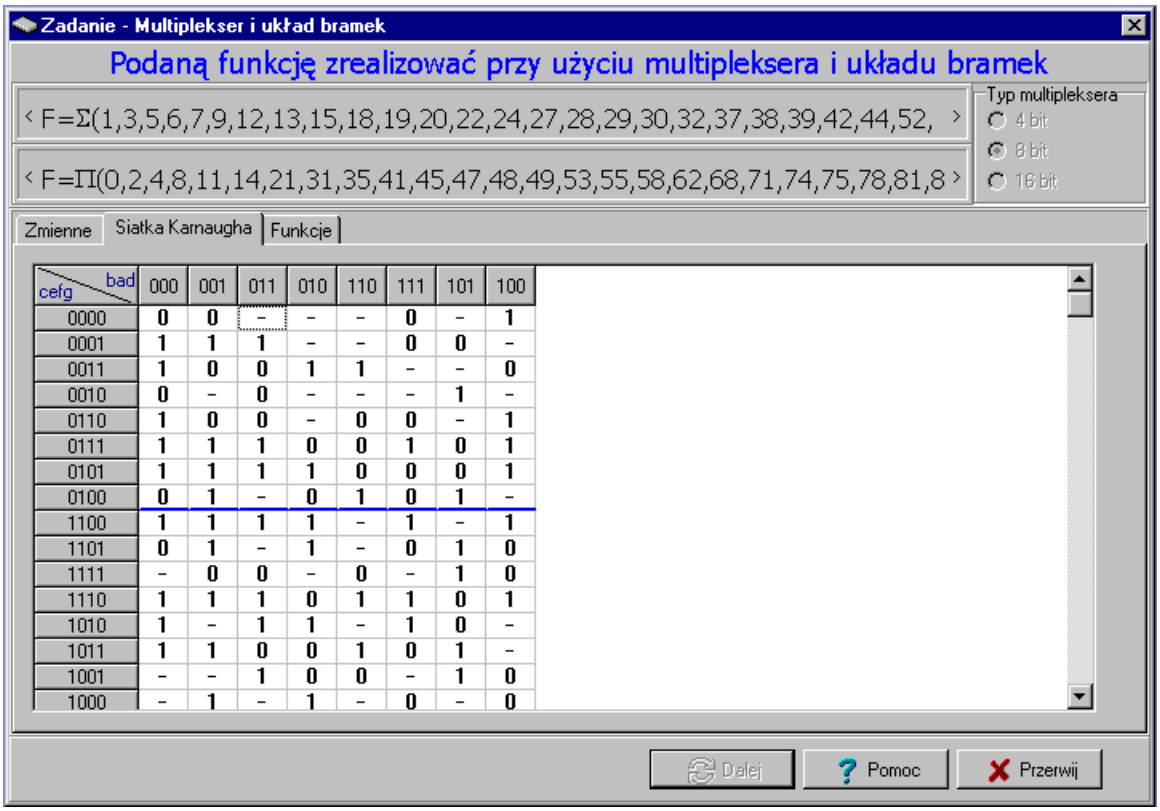
Wybór polecenia „Multiplekser i układ bramek” umożliwia przejście do syntezy i realizacji układu. W pierwszym etapie użytkownik musi dokonać podziału zmiennych na podstawowe i dodatkowe. Celem ustalenia właściwej kolejności zmiennych konieczne jest przeprowadzenie minimalizacji realizowanej funkcji. Podgląd postaci minimalnej sumy i iloczynu dostępny jest po wciśnięciu przycisków „Pomoc” i „Pokaż funkcje zminimalizowane”, które umieszczono na zakładce „Zmienne” (rys. 4).



Rys. 4. Okno MDI programu z opcją zadanie
Fig. 4. MDI Windows with task option

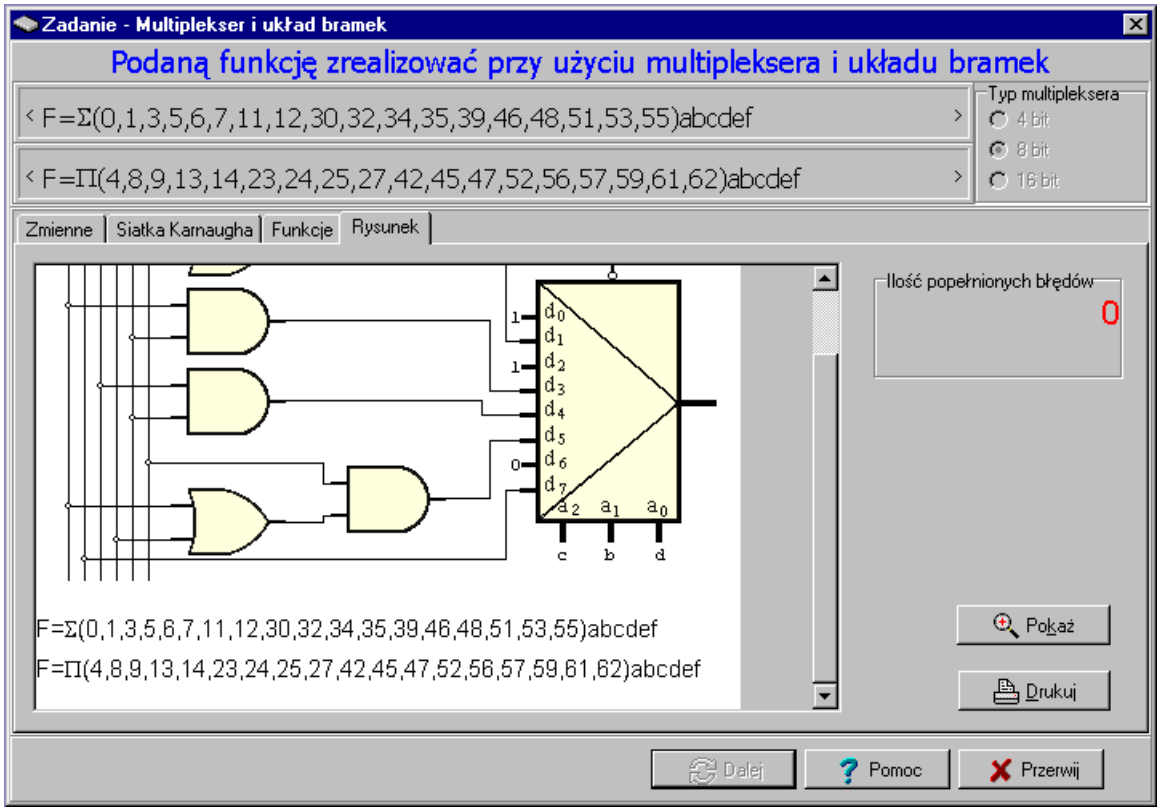
W przypadku gdy użytkownik wprowadzi poprawne zmienne, ale nie zachowa kolejności alfabetycznej, wówczas na ekranie pojawi się okienko informacyjne. Algorytm zaimplementowany w programie sprawdza wprowadzony podział zmiennych podstawowych i dodatkowych oraz kolejność ich alfabetyczną. W przypadku wprowadzenia poprawnego podziału zmiennych podstawowych i dodatkowych z zachowaniem porządku alfabetycznego komunikat ten jest pomijany. W kolejnym etapie program wygeneruje siatkę Karnaugha opisaną za pomocą zmiennych podstawowych i dodatkowych. Na podstawie wygenerowanej siatki (rys. 5) użytkownik ustala funkcje wejść dla multipleksera. Aby wyedytować funkcje wejść,

należy uruchomić edytor funkcji, dostępny po naciśnięciu przycisku „Edytuj funkcję” na zakładce „Funkcje” (rys. 5). Dla wejścia multipleksera wybranego na zakładce „Funkcje” tworzone jest wyrażenie minimalne dla funkcji tego wejścia za pomocą „Edytora funkcji”. Edytor posiada szereg przycisków ułatwiających tworzenie wyrażenia. Wygenerowana postać wyrażenia pojawia się na górnym panelu edytora. Ostateczną jego postać zatwierdza się po naciśnięciu przycisku „Zatwierdź”.



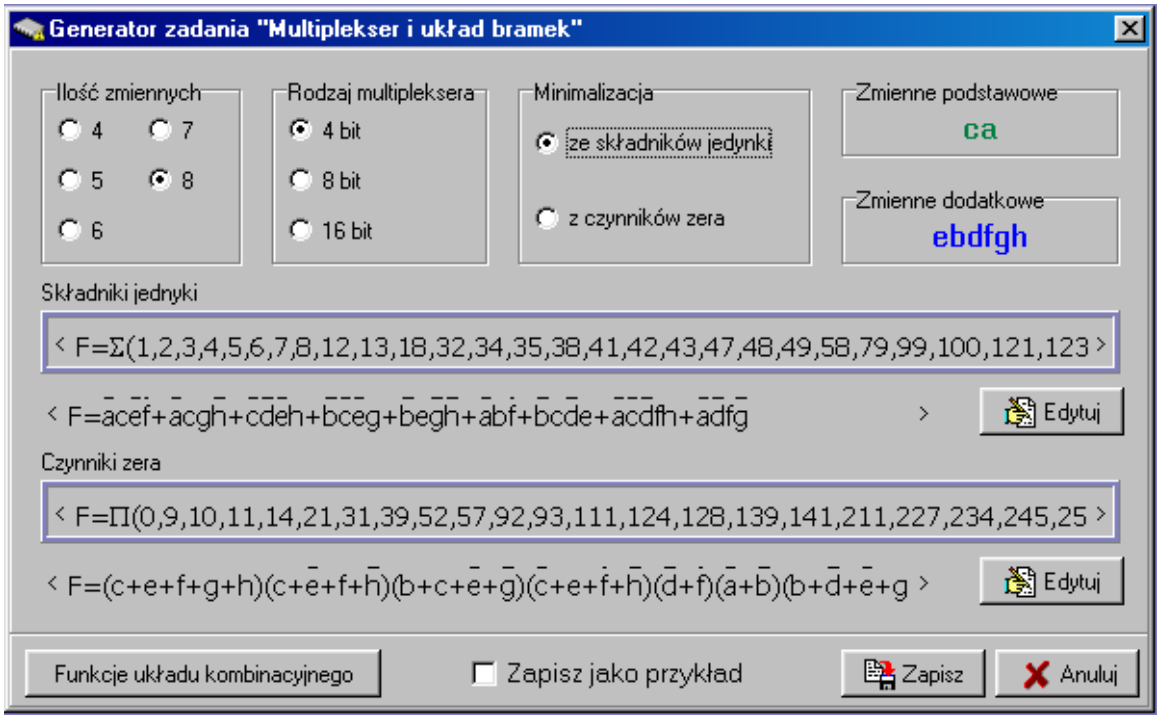
Rys. 5. Okno programu z siatką Karnaugha
Fig. 5. Windows with Karnaugh map

W przypadku gdy użytkownik wprowadzi błędne wyrażenie, zostanie wygenerowany komunikat o błędzie. Program automatycznie usunie błędnie wprowadzone dane, pozostawiając tylko poprawne wyrażenia. Po wprowadzeniu wszystkich poprawnych danych i wciśnięciu przycisku „Dalej” nastąpi wygenerowanie schematu logicznego układu na zakładce „Rysunek” (rys. 6). Wydrukowanie schematu dostępne jest za pomocą przycisku "Drukuj". Na zakładce dostępna jest także opcja „Pokaż”, wyświetlająca schemat logiczny w osobnym oknie, a także wyświetlany jest komunikat o liczbie popełnionych błędów podczas rozwiązywania zadania.

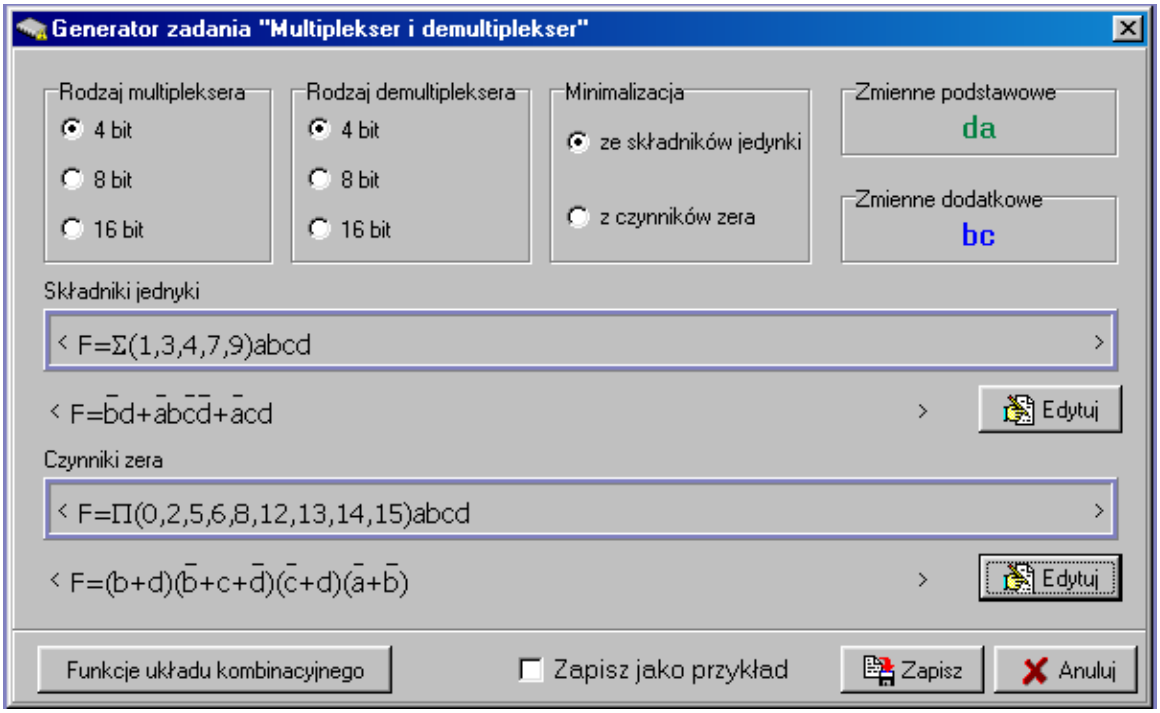


Rys. 6. Okno MDI z wygenerowanym układem bramek i multiplekserem
Fig. 6. MID Windows with generated schematic diagram of gates and multiplexer

Wybór polecenia „Multiplekser i układ bramek” w opcji Generator zadań spowoduje, że na ekranie pojawi się okno o wyglądzie jak na rys. 7. W celu stworzenia pliku zadania lub pliku przykładu dla struktury „Multiplekser i układ bramek” w pierwszej kolejności należy wskazać rozmiar multipleksera (4-, 8- lub 16-bitowy), określić liczbę zmiennych (4-8) oraz rodzaj minimalizacji (ze składników jedynki lub z czynników zera). Po ustaleniu powyższych danych należy wprowadzić składniki jedynki oraz czynniki zera funkcji logicznej w zapisie dziesiętnym. Możliwe jest także wprowadzanie danych w zapisie binarnym, po uprzednim zaznaczeniu opcji „Wprowadzaj binarnie” w edytorze funkcji. Przyciski „Edytuj” znajdujące się na formularzu Generator zadań – Multiplekser i układ bramek uruchamiają edytor zbioru składników jedynki lub czynników zera. Po wprowadzeniu danych funkcji w Generatorze zadań program automatycznie wygeneruje funkcje wejścia dla struktury „Multiplekser i układu bramek”. Wygenerowane funkcje dostępne są na formularzu „Funkcje układu kombinacyjnego”, który uruchamiany jest przyciskiem znajdującym się w generatorze zadań. Funkcje te nie są zapisywane do pliku z zadaniem lub przykładem, lecz będą wygenerowane automatycznie przez program podczas rozwiązywania zadania lub przykładu. Pozwala to na zabezpieczenie dostępu do danych w programie.



Rys. 7. Okno programu generatora zadań
Fig. 7. Windows of tasks generator



Rys. 8. Okno programu generatora zadań
Fig. 8. Windows with tasks generator

Wybór polecenia „Multiplekser i Demultiplekser” w opcji Generator zadań spowoduje, że na ekranie pojawi się okno o wyglądzie jak na rysunku 7. W celu stworzenia pliku zadania lub pliku przykładu dla struktury „Multiplekser i Demultiplekser” należy wskazać rozmiar multipleksera (4-, 8- lub 16-bitowy) oraz demultipleksera (4-, 8- lub 16-bitowy), a następnie

określić rodzaj minimalizacji, tj. ze składników jedynki lub z czynników zera. Po ustaleniu powyższych danych należy wprowadzić składniki jedynki oraz czynniki zera opisujące funkcję logiczną za pomocą edytorów dostępnych po użyciu odpowiedniego przycisku „Edytuj". Sposób postępowania podczas tworzenia pliku zadania lub pliku przykładu jest raczej intuicyjny i nie powinien stwarzać użytkownikowi problemów. Wygenerowany plik dla zadania dotyczącego tej struktury zostanie zapisany z rozszerzeniem *.md. Pliki z danymi dla przykładu zostaną zapisane z rozszerzeniem *.emd.

4. Podsumowanie

Opracowany program umożliwia wspomaganie nauczania z zakresu komutatorów. Możliwe jest wprowadzanie danych dla funkcji logicznej w postaci dziesiętnej i binarnej. W celu uzyskania optymalnego rozwiązania układowego przeprowadzana jest minimalizacja wyrażenia dla wprowadzonej funkcji logicznej. Wykonanie minimalizacji wyrażenia przeprowadzane jest zarówno dla składników jedynki, jak i czynników zera. Wynikiem jest funkcja w postaci wyrażenia minimalnego sumy iloczynów i iloczynu sum. Opracowany pogram umożliwia zapoznanie się zagadnieniami teoretycznymi z zakresu komutatorów, rozwiązywanie zadań przez użytkownika z komputerową kontrolą kolejnych kroków, tworzenie danych dla prezentacji i dla zadań, prezentację syntezy w oparciu o cztery struktury z możliwością wyboru rozmiaru elementów i liczby zmiennych, a schematy logiczne otrzymane w prezentacjach i różnorodnych zadaniach mogą być wydrukowane wraz z opisem realizowanej funkcji. Podsumowując, można stwierdzić, że program pozwala na demonstrację przykładów, sprawdzenie wiedzy dla różnych danych, które nie są narzucone („zaszyte w programie"), ale mogą być wybierane przez użytkownika zależnie od poziomu zdobytej wiedzy.

LITERATURA

1. Małysiak H. – red.: Teoria automatów cyfrowych – laboratorium. Skrypt Pol. Śl., Gliwice 1994.
2. Xiao Y., He S.: Optics Communications, Vol. 239, 1-3, September 2004.
3. Jong-Ru G., Chao Y., Kuan Z., Michael C., Curran P., Jiedong D., Bryan G., Kraft R., McDonald J.: Integration the VLSI Journal, Vol 38, 3 January 2005.
4. Kamionka-Mikuła H.: Komputerowe wspomaganie nauczania w zakresie układów synchronicznych. Pol. Śl. Studia Informatica Vol. 22, No. 3(45), Gliwice 2001.

5. Małysiak H., Bonk M.: Zdalne wspomaganie nauczania na przykładzie budowy i działania mikroprocesorów IA-64. Pol. Śl. Studia Informatica Vol. 24, No 2A(53), Gliwice 2003.
6. Madej K., Masarek K., Kuryłowicz K.: Komputery osobiste. WKŁ, Warszawa 1981.
7. Kamionka-Mikuła H., Małysiak H., Pochopień B.: Układy cyfrowe – teoria i przykłady. WPK J. Skalmierskiego, Gliwice 2000.
8. Kamionka-Mikuła H.: Materiały dydaktyczne – Teoria automatów cyfrowych. Pol. Śl. Instytut Informatyki, Gliwice 1999.

Recenzent: Dr inż. Halina Kamionka-Mikuła

Wpłynęło do Redakcji 25 kwietnia 2005 r.

Abstract

The article describes an example of a solution for system aided teaching of commutators. Possibilities and versatility of exercises of didactic program used for teaching of multiplexers and demultiplexers have been presented. In the program there has been made an assumption that the user will make a decision about a kind of multiplexer and demultiplexer (4, 8 or 16-bits) and also about a number of variables for any structure realization. Logical functions can be realized in single multiplexer and single demultiplexer structures. However, logical functions with the number of arguments $p \leq 4$ can be accepted too. In this case they can be realized in the structure of multiplexer and logical gates or in the structure of multiplexer and demultiplexer. The application presented allows for resolving of problematic tasks with different level of difficulty, examples tasks and also user generated tasks.

Adres

Dariusz STRZĘCIWILK: Szkoła Główna Gospodarstwa Wiejskiego, Katedra Ekonometrii i Informatyki, ul. Nowoursynowska 161, 02-787 Warszawa, Polska, dstrzeciwilk@mors.sggw.waw.pl