ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1995

Seria: INFORMATYKA z.29 Nr kol. 1306

Anna TOMASZEWSKA

OPROGRAMOWANIE WSPOMAGAJĄCE NAUCZANIE ZAGADNIEŃ ZESPOŁÓW FPLS

Streszczenie. Struktura programowanych zespołów logicznych FPLS doskonale odzwierciedla strukturę asynchronicznych układów sekwencyjnych, co pozwala w prosty sposób realizować za ich pomocą charakterystyczne dla tych układów funkcje stanu. W artykule prezentowane jest oprogramowanie wspomagające nauczanie, obejmujące tematykę związaną z projektowaniem asynchronicznych układów z pamięcią i ich realizacją sprzętową z wykorzystaniem zespołów FPLS.

COMPUTER-ASSISTED INSTRUCTION SOFTWARE FOR FPLS UNITS PROBLEMS

Summary. This article presents a computer-assisted instruction software concerning Field Programmable Logic Sequencers' problems. FPLS units, due to their structure, let designers implement in a simple way the state functions which are characteristic for the asynchronous sequential switching circuits. That is why the software system described in this paper relates mainly to designing and hardware implementation of such circuits with use of FPLS.

DIE SOFTWARE ZUM RECHNERGESTÜTZTEN LERNEN IM BEREICH DER PROBLEME DER FPLS–BLÖCKE

Zusammenfassung. Die Struktur programmierter logischer FPLS–Blöcke widerspiegelt vorzüglich die Struktur asynchroner sequentieller Systeme, was erlaub, für sie charakteristische Zustandsfunktionen mit Hilfe der FPLS-Blöcke auf einfache Weise zu realisieren. Im Artikel wird die Software zum rechnergestützten Lernen dargestellt, die die Entwurfsprobleme asynchroner Systeme mit Speicher und ihrer Hardware-Realisierung mit Ausnutzung der FPLS-Blöcke umfaßt.

1. Wprowadzenie

Rozwój techniki w ostatnich dziesięcioleciach znalazł swoje odzwierciedlenie m. in. w zmianie sposobu nauczania. Coraz częściej w procesie przekazywania wiedzy korzysta się z dostępnych środków audiowizualnych. Ma to na celu zwiększenie efektywności nauczania dzięki urozmaiceniu prowadzonych zajęć. Tradycyjne metody, których podstawą są wykłady ilustrowane rysunkami na tablicy czy samodzielne studiowanie podręczników, wzbogacone zostają przez ruchomy, barwny obraz oferowany przez telewizję. Dzięki temu zjawiska występujące w makro- lub mikroświecie przestają być dla nas czymś, co możemy jedynie wyobrazić sobie na podstawie istniejących opisów.

Jedną z najnowszych pomocy naukowych, z której coraz częściej korzysta się w szkolnictwie jest komputer. Wraz z konstruowaniem coraz lepszego sprzętu pojawia się również coraz więcej oprogramowania, między innymi również stanowiącego pomoc dydaktyczną. Tworzą się zespoły programistów pracujące nad niejednokrotnie bardzo złożonymi systemami edukacyjnymi. O tym, jak bardzo zainteresowano się tym tematem, świadczy także fakt, iż spośród nauk humanistycznych wyodrębniła się nowa dziedzina, zajmująca się podstawami teoretycznymi, zastosowaniami i metodami nauczania wspomaganego komputerem (ang. *Computer Assisted Instruction Software - CAIS*).

Wśród programów zaliczanych do oprogramowania edukacyjnego wyróżnia się kilka grup, w zależności od tego, w jaki sposób pośredniczą one w przekazywaniu wiedzy. Podstawowe grupy scharakteryzowano krótko poniżej.

Programy demonstracyjne i symulatory – obrazują one przebieg zjawisk, których przedstawienie w naturze jest niemożliwe lub utrudnione ze względu na skalę, w jakiej występują, warunki, czas trwania lub koszty, bądź też realizują, w sposób dokładny lub przybliżony, system opisany pewnym modelem teoretycznym.

Programy ukierunkowane na ćwiczenia i doskonalenie umiejętności – pomagają one w pamięciowym opanowaniu pewnych pojęć i czynności, utrwaleniu pewnych metod postępowania, zapamiętaniu kolejności wykonywanych operacji przy rozwiązywaniu zadań według danego algorytmu. Programy te reagują na dane wprowadzane przez użytkownika z klawiatury i informują go za pośrednictwem komunikatów o stanie jego wiedzy i umiejętności.

Programy typu "korepetytor" – charakteryzują się one tym, że nauczany materiał podzielony jest na fragmenty zróżnicowane pod względem trudności i/lub stopnia abstrakcji. Użytkownik korzystając z programu zapoznaje się na początku z najłatwiejszymi zagadnieniam. Po przerobieniu danego fragmentu teoretycznie, uczący się wykonuje ćwiczenia praktyczne, ewentualnie odpowiada na pytania testowe. Stopień poprawności odpowiedzi kwalifikuje go do

przejścia do następnej partii materiału lub informacji i zadań uzupełniających. Stwarza to możliwość indywidualizacji procesu nauczania.

Nie każdy program dydaktyczny należy do którejś z opisanych wyżej grup. Przeważnie systemy wspomagające nauczanie łączą w sobie cechy charakterystyczne dla grupy programów demonstracyjnych, uczących przez ćwiczenia i sprawdzających wiedzę użytkownika w stopniu zależnym od rodzaju prezentowanych zagadnień.

Podobnie napisany został program przedstawiony w tej publikacji. Składa się on z trzech części, z których każda pozwala użytkownikowi w inny sposób zapoznawać się z zagadnieniem projektowania asynchronicznych układów sekwencyjnych z zastosowaniem zespołów FPLS.

2. Zagadnienia wybrane do implementacji

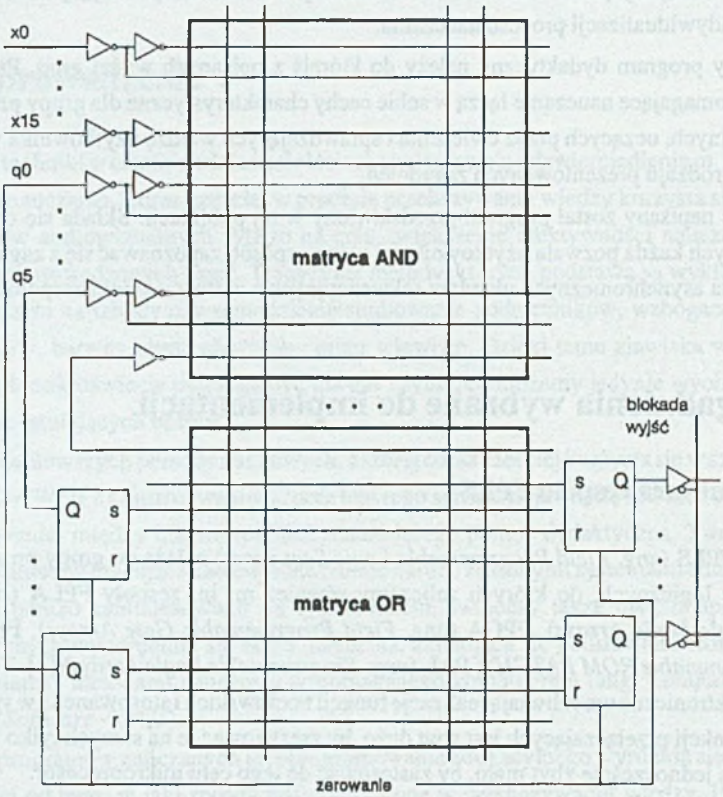
2.1. Struktura zespołu FPLS

Układy FPLS (*ang. Field Programmable Logic Sequencer*) należą do grupy programowanych matryc logicznych, do których zaliczamy również m. in. zespoły FPLA (*ang. Field Programmable Logic Arrays*), FPGA (*ang. Field Programmable Gate Arrays*), FPRP (*ang. Field Programmable ROM PATCH*) i PAL (*ang. Programmable Logic Arrays*) [4]. Te cyfrowe elementy elektroniczne umożliwiają realizację funkcji boolowskich i stosowane są w systemach, w których funkcji przełączających jest zbyt dużo, by zrealizować je na samych tylko bramkach logicznych, a jednocześnie zbyt mało, by zastosować do tego celu mikroprocesor.

Rysunek 1 obrazuje strukturę blokową typowego zespołu FPLS (82S104 firmy Signetics). Zespół ma 16 wejść i 8 wyjść. W skład jego wchodzą następujące moduły:

— wejściowe układy sterujące, umożliwiające korzystanie z sygnałów wejściowych (x0 - x15) w postaci prostej lub zanegowanej,
— matryca programowanych iloczynów (48 bramek AND 45-wejściowych),
— matryca programowanych sum (28 bramek OR 48-wejściowych),
— 14 przerzutników asynchronicznych sr, z których 6 stanowi rejestr stanu (sygnały q0 - q5), a 8 – rejestr wyjściowy (sygnały y0 - y7),
— wyjściowe układy separujące (bufory 3-stanowe).

Sygnały wyjściowe uzyskuje się z sygnałów dostępnych w matrycy iloczynów przez wyznaczenie połączeń między odpowiednimi liniami w matrycach AND i OR. Zależności funkcyjne opisujące sygnały podawane na wejścia przerzutników muszą mieć postać sum iloczynów, nie stanowi to jednak problemu dla projektanta systemu, ponieważ każdą funkcję logiczną można przedstawić w tej postaci.



Rys. 1. Struktura zespołu FPLS (Signetics 82S104)
Fig. 1 The structure of FPLS unit (Signetics 82S104)

Analizując strukturę zespołu zauważyć należy, że sygnały z wyjść przerzutników stanowiących rejestr stanu zawracane są do matrycy iloczynów, gdzie dostępne są w postaci prostej i zanegowanej. Powstałe w ten sposób sprzężenie zwrotne daje układowi możliwości odróżniające go zasadniczo od pozostałych zespołów zawierających programowane matryce logiczne. Umożliwia mianowicie realizację funkcji przełączających, których wartość w chwili bieżącej (t) jest zależna od ich wartości w chwili poprzedniej ($t - \tau$), tak zwanych funkcji sekwencyjnych. Pozostałe z wymienionych wcześniej zespołów realizują tylko funkcje kombinacyjne.

Cyfrowe układy sekwencyjne (wielotaktowe) charakteryzują się tym, że istnieje przynajmniej jeden taki stan wejść układu, któremu odpowiada kilka różnych stanów wyjść. Aktualny stan wyjść zależy nie tylko od stanu wejść układu, ale również od stanu wewnętrznego zależnego od stanu sygnałów wejściowych w chwilach poprzednich. Z tego względu układy sekwencyjne nazywane są automatami z pamięcią [5]. Stan wewnętrzny automatu sekwencyjnego opisują funkcje stanu (zwane również funkcjami przejść) będące typowymi funkcjami sekwencyjnymi.

Jeżeli dla funkcji stanu asynchronicznego automatu sekwencyjnego wyznaczymy funkcje wzbudzeń przerzutników *sr*, to możemy automat taki zrealizować w prosty sposób korzystając z typowego programowanego zespołu FPLS, traktując jego rejestr stanu jako blok pamięci generujący sygnały stanu.

2.2. Funkcje opisujące automat sekwencyjny

Asynchroniczne układy sekwencyjne opisywane są przez dwa rodzaje funkcji: funkcje stanu (przejść) i funkcje wyjść. Rysunek 2 przedstawia dwa typy układów wielotaktowych: automat Mealy'ego i automat Moore'a. Jeżeli wektory sygnałów oznaczymy:

$Q = \{q_1, q_2, ..., q_k\}$ – wektor sygnałów stanu,
$X = \{x_1, x_2, ..., x_m\}$ – wektor sygnałów wejściowych, (2.2.1)
$Y = \{y_1, y_2, ..., y_n\}$ – wektor sygnałów wyjściowych,

to zależności opisujące stan wewnętrzny można przedstawić wzorem:

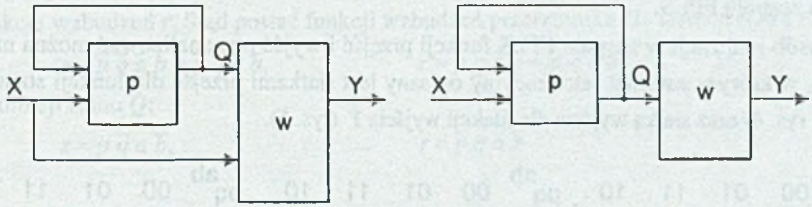
$$Q(t) = p(Q(t-\tau), X(t)) \quad (2.2.2)$$

Funkcje wyjść natomiast mogą być zależne od wektora stanu i wektora wejściowego:

$Y(t) = w(Q(t), X(t))$ – mamy wówczas do czynienia z tzw. automatem Mealy'ego

lub tylko od wektora stanu:

$Y(t) = w(Q(t))$ – w przypadku automatu Moore'a.



Rys. 2. Rodzaje automatów sekwencyjnych: automat Mealy'ego i automat Moore'a
Fig. 2.Types of sequential automata: Mealy automaton and Moore automaton

Powyższe zależności funkcyjne dla asynchronicznego układu sekwencyjnego wyznaczyć można na podstawie rozwiązalnej tablicy kolejności łączeń bądź siatek przejść/wyjść. Oprogramowanie prezentujące pierwszy ze sposobów już istnieje [1]. W prezentowanej pracy zdecydowano się więc na przedstawienie drugiej drogi rozwiązania. Za wyborem tym dodatkowo przemawiał fakt, iż niejako przy okazji zaznajomić można użytkownika programu z metodą pozwalającą bezpośrednio z siatek stanu w prosty sposób wyznaczyć funkcje wzbudzeń przerzutników, stanowiących elementy pamięci układu. W dalszej części metoda ta nazywana będzie metodą "wyróżnionych zer i jedynek".

2.3. Wyznaczanie i realizacja funkcji przejść i funkcji wyjść – przykład

W skład zespołów FPLS prezentowanych w programie wchodzą przerzutniki asynchroniczne *sr*, za pomocą których realizuje się zarówno funkcje przejść dla automatu sekwencyjnego, jak i funkcje wyjść. Aby zrealizować za pomocą przerzutnika dowolną funkcję elementu pamięci, trzeba określić funkcje sterujące dla jego wejść.

Dla funkcji sekwencyjnych opracowano do tej pory trzy metody: algebraiczną, transformacji wyznaczonej wcześniej funkcji logicznej elementu pamięci i wyznaczania funkcji wzbudzeń bezpośrednio na podstawie siatki przejść i tablicy wzbudzeń przerzutnika [5]. Ostatnia z tych metod uległa modyfikacji - uproszczeniu. Stała się ona dzięki temu bardzo wygodna i szybka. Najlepiej też nadaje się ona do implementacji komputerowej. Metodę zmodyfikowaną opublikowano po raz pierwszy w opracowaniu [3], a wykorzystywano już wcześniej w pracach [8]. Mając dane siatki stanu automatu otrzymane z wolnej od wyścigu krytycznego tablicy programu można, omijając etap wyznaczania funkcji stanu, od razu podać postać odpowiadających im funkcji sterujących przerzutników stanowiących elementy pamięci układu. Korzysta się przy tym z faktu, iż wystarczy, aby wejścia przerzutników pobudzone były tylko w stanach niestabilnych układu.

Funkcje wyjść są funkcjami kombinacyjnymi – stany wyjść układu w chwili $(t+\tau)$ nie zależą od stanu wyjść w chwili (t) – nie da się więc określać dla nich funkcji wejść *s* i *r* przerzutnika za pomocą metody "wyróżnionych zer i jedynek". W tym przypadku posłużyć się jednak można tradycyjną metodą grupowania sąsiednich logicznie pól siatek Karnaugha i wykorzystać właściwości zespołu FPLS.

Sposób realizacji w zespole FPLS funkcji przejść i wyjść przeanalizować można na przykładzie, w którym automat sekwencyjny opisany jest siatkami przejść dla funkcji stanu *P* i *Q* (rys. 3 i rys. 4) oraz siatką wyjścia dla funkcji wyjścia *Y* (rys. 5).

pq \ ab	00	01	11	10
00	0	1	–	1
01	0	0	–	0
11	–	0	–	1
10	0	1	–	1

P

Rys. 3. Siatka przejść dla funkcji stanu *P*
Fig. 3. Next-state K-map for *P* state function

pq \ ab	00	01	11	10
00	0	0	–	0
01	0	1	–	1
11	–	1	–	1
10	0	0	–	1

Q

Rys. 4. Siatka przejść dla funkcji stanu *Q*
Fig. 4. Next-state K-map for *Q* state function

pq \ ab	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	0	1	1	0
10	0	0	1	0

Y

Rys. 5. Siatka wyjścia
Fig. 5. Output K-map

Proces wyznaczania i realizacji funkcji obejmuje trzy opisane w kolejnych podpunktach etapy.

2.3.1. Wyznaczanie funkcji przejść

W siatkach przejść poszukuje się kratek (pól) odpowiadających stanom niestabilnym układu, czyli takich, dla których wartość bieżąca zmiennej stanu różni się od jej wartości następnej. Kratki takie wyróżnia się przez pogrubienie wpisanych w nich wartości (rys. 6 i rys. 7).

pq \ ab	00	01	11	10
00	0	**1**	–	1
01	0	0	–	0
11	–	**0**	–	1
10	**0**	1	–	1

P

Rys. 6. Siatka przejść dla funkcji stanu P
Fig. 6. Next-state K-map for P state function

pq \ ab	00	01	11	10
00	0	0	–	0
01	**0**	1	–	1
11	–	1	–	1
10	0	0	–	**1**

Q

Rys. 7. Siatka przejść dla funkcji stanu Q
Fig. 7. Next-state K-map for Q state function

Pola siatki z wyróżnionymi jedynkami to te i tylko te, dla których wartość sygnału s musi wynosić 1, w związku z tym kombinacje sygnałów im odpowiadające wchodzą jako składniki sumy do wyrażenia na funkcję wzbudzeń s. Podobnie kombinacje sygnałów odpowiadające polom zawierającym wyróżnione zera wchodzą jako składniki sumy do wyrażenia na funkcję wzbudzeń r. Stąd postać funkcji wzbudzeń przerzutnika dla funkcji stanu P:

$$s = \bar{p}\,\bar{q}\,\bar{a}\,b + \bar{p}\,\bar{q}\,a\,\bar{b}, \qquad r = p\,q\,\bar{a}\,b + p\,\bar{q}\,\bar{a}\,\bar{b}$$

i dla funkcji stanu Q:

$$s = p\,\bar{q}\,a\,\bar{b}, \qquad r = \bar{p}\,q\,\bar{a}\,\bar{b}$$

2.3.2. Wyznaczanie funkcji wyjścia

pq \ ab	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	0	1	1	0
10	0	0	1	0

Y

Funkcję kombinacyjną w postaci sumy iloczynów określającą sygnał wyjściowy Y wyznaczyć można tradycyjnie metodą grupowania sąsiednich pól siatek Karnaugha w siatce wyjść (rys.8). Stąd

$$Y = \bar{p}\,\bar{a} + p\,q\,b + p\,a\,b$$

Rys. 8. Siatka wyjścia
Fig. 8. Output K-map

2.3.3. Realizacja funkcji w matrycach

Gdy wyznaczone są już funkcje sterujące przerzutników, należy każdy z iloczynów występujących w wyrażeniach zrealizować na osobnej linii matrycy AND. Pojedynczy iloczyn realizuje się w ten sposób, że na jednej z linii pionowych matrycy iloczynów zaznacza się połączenia między tą linią a liniami doprowadzającymi sygnały wejściowe i stanu odpowiadające literałom występującym w danym iloczynie. W przypadku występowania wyścigu niekrytycznego ten sam składnik może się znaleźć w dwóch lub więcej wyrażeniach opisujących funkcje sterujące przerzutników. Wystarczy wtedy zrealizować go tylko na jednej linii matrycy AND. Po zaznaczeniu połączeń dla iloczynów trzeba wypełnić matrycę sum.

Dla funkcji stanu wypełnianie matrycy OR polega na tym, że na liniach prowadzących do wejść *s* i *r* poszczególnych przerzutników, stanowiących rejestr stanu, zaznacza się połączenia z liniami matrycy AND odpowiadającymi iloczynom wchodzącym w skład sumy opisującej daną funkcję sterującą dla *s* lub *r*. Na rysunku 9 pokazano przykład realizacji wyznaczonych wcześniej funkcji sterujących przerzutników. Na linii prowadzącej do wejścia *s* przerzutnika *P* zsumowano iloczyny zrealizowane na liniach 1 i 2 matrycy AND, a na linii prowadzącej do wejścia *r* przerzutnika *P* - iloczyny z linii 3 i 4. Podobnie linię prowadzącą do wejścia *s* przerzutnika *Q* połączono z linią 5, a linię dla wejścia *r* tego przerzutnika z linią 6 matrycy iloczynów. Wartości sygnałów generowanych na wyjściach przerzutników stanowiących rejestr stanu będą zgodne z wartościami wpisanymi w pola siatki przejść.

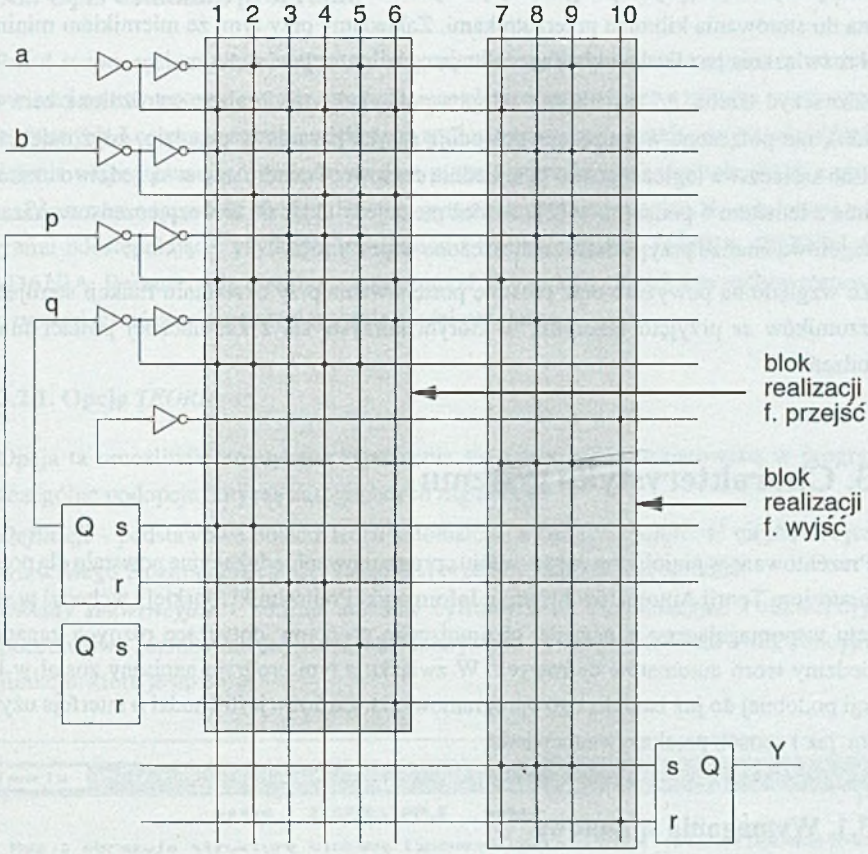
Inaczej realizuje się funkcję wyjścia. Jeżeli projektowany układ ma tylko jedno wyjście, to w prosty sposób można wypracować sygnał dla niego korzystając z dodatkowej linii matrycy OR zawracanej do matrycy AND przez negację. Linie matrycy iloczynów odpowiadające składnikom sumy dla funkcji *Y* (tzn. linie 7, 8 i 9) łączy się w matrycy sum zarówno z linią prowadzącą do wejścia *s* przerzutnika *Y*, jak i z linią zawracaną przez negację do matrycy AND. Z kolei sygnał ten, będący w matrycy iloczynów negacją wartości *Y*, podaje się na wejście *r* przerzutnika *Y* za pośrednictwem połączeń na jednej z linii matrycy iloczynów (linia 10). Połączenie takie zapewnia poprawną realizację funkcji wyjścia.

W tabeli 1 podano, na których liniach matrycy AND zrealizowano poszczególne iloczyny.

Tabela 1

Realizacja iloczynów na liniach matrycy AND

nr linii	iloczyn	nr linii	iloczyn	nr linii	iloczyn
1	$\bar{p}\,\bar{q}\,\bar{a}\,b$	4	$p\,\bar{q}\,\bar{a}\,\bar{b}$	7	$\bar{p}\,\bar{a}$
2	$\bar{p}\,\bar{q}\,a\,\bar{b}$	5	$p\,\bar{q}\,a\,\bar{b}$	8	$p\,q\,b$
3	$p\,q\,\bar{a}\,b$	6	$\bar{p}\,q\,\bar{a}\,\bar{b}$	9	$p\,a\,b$



Rys. 9. Realizacja funkcji przejść i funkcji wyjścia w zespole FPLS
Fig. 9. Realization of state-functions and output-function in FPLS unit

2.3.4. Uwagi do prezentowanej metody

W siatkach z pogrubionymi zerami i jedynkami można również tworzyć grupy w celu zminimalizowania wyrażenia dla funkcji wzbudzeń. Pogrubione jedynki (zera) grupować można z jedynkami (zerami) nie wyróżnionymi, oznaczającymi stany stabilne, oraz ze stanami nie określonymi. Ważne jest to, aby wszystkie wyróżnione jedynki (zera) pokryte były grupami. W programie przedstawionym w niniejszym artykule stosowana jest opisana wyżej metoda w wersji bez tworzenia grup w celu minimalizacji wyrażenia. Określanie postaci minimalnej nie zawsze bowiem prowadzi do rozwiązania minimalnego. Przykładem może być przypadek, gdy

w siatce przejść występuje wyścig niekrytyczny. Wówczas ten sam pełny iloczyn wykorzystać można do sterowania kilkoma przerzutnikami. Zakładamy przy tym, że miernikiem minimalności rozwiązania jest liczba układów realizujących iloczyny.

Zaznaczyć trzeba, że sąsiadujące ze sobą pola siatki zawierające wyróżnione zera (lub jedynki), nie połączone w grupę, nie powodują ryzyka hazardu w układzie, gdyż najczęściej pomimo sąsiedztwa logicznego nie są sąsiednie czasowo. Ponadto nawet sąsiedztwo czasowe, zgodnie z lematem 5 podanym w [2] również nie daje w układzie niebezpieczeństwa hazardu. Szczegółową analizę przypadków zamieszczono w pracy [6].

Ze względu na powyższe oraz prostotę postępowania przy określaniu funkcji sterujących przerzutników *sr* przyjęto algorytm, w którym korzysta się z kanonicznej postaci funkcji wzbudzeń.

3. Charakterystyka systemu

Prezentowane w niniejszym opracowaniu oprogramowanie edukacyjne powstało dla potrzeb Laboratorium Teorii Automatów Instytutu Informatyki Politechniki Śląskiej i wchodzi w skład pakietu wspomagającego nauczanie, obejmującego programy dotyczące różnych zagadnień z dziedziny teorii automatów cyfrowych. W związku z tym program napisany został w konwencji podobnej do już istniejącego oprogramowania, zarówno jeśli chodzi o interfejs użytkownika jak i sposób przekazywania wiedzy.

3.1. Wymagania sprzętowe

System napisany został dla dowolnego komputera klasy IBM PC pracującego w środowisku DOS. Nie jest wymagana większa pojemność pamięci operacyjnej niż 640 kB. Karty graficzne, z którymi współpracuje program, to:

— Hercules,
— EGA (z monitorem monochromatycznym lub kolorowym),
— VGA lub SVGA (z monitorem monochromatycznym lub kolorowym).

Rozmiar kodu programu: 118 kB. Dane zajmują w PaO 41 kB. Ponadto oprogramowanie wymaga ok. 20 kB wolnej przestrzeni w pamięci operacyjnej, jeżeli jest uruchamiany na komputerze z kartą graficzną Hercules lub 160 kB w przypadku pozostałych kart. Z obszaru tego korzysta się w celu przechowywania fragmentów ekranu graficznego zapamiętywanych i odtwarzanych w czasie pracy z programem.

3.2. Opis działania programu

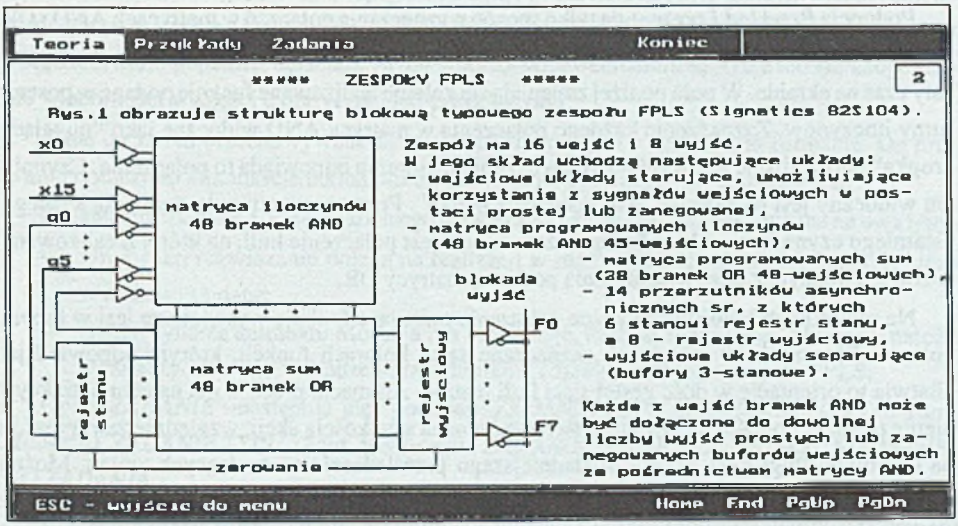
Przed rozpoczęciem właściwej pracy z systemem użytkownik proszony jest o udzielenie odpowiedzi na trzy testowe pytania, sprawdzające elementarną wiedzę z zakresu teorii automatów cyfrowych. Losowane są one z większego zestawu, co zapewnia małą powtarzalność trójek. Udzielenie niepoprawnej odpowiedzi na dwa pytania powoduje zakończenie pracy z programem. Poprawna odpowiedź na dwa lub trzy pytania zapewnia przejście do właściwej części programu udostępniającej użytkownikowi menu z trzema opcjami: TEORIA, PRZYKŁADY i ZADANIA. Reprezentują one trzy niezależne od siebie, jeśli chodzi o przepływ sterowania między nimi, choć powiązane tematycznie części programu.

3.2.1. Opcja *TEORIA*

Opcja ta umożliwia teoretyczne zapoznanie się z tematyką prezentowaną w programie. Poszczególne podopcje dotyczą następujących zagadnień:

Definicje - podstawowe pojęcia teorii automatów, a których znajomość niezbędna jest do właściwego zrozumienia całości zagadnień przedstawianych w programie;

Układy sekwencyjne - podział układów cyfrowych na kombinacyjne i sekwencyjne - podstawowe różnice między nimi, charakterystyka i rodzaje automatów sekwencyjnych, funkcje, które je opisują;



Rys.10. Wygląd ekranu w opcji Teoria/Zespoły FPLS
Fig.10. Image of the screen in Teoria/Zespoły FPLS option

Zespoły FPLS - rodzaje programowanych układów logicznych, opis struktury typowego zespołu FPLS z uwzględnieniem podziału na matryce iloczynów i sum, zasada działania, zastosowanie do realizacji asynchronicznych układów sekwencyjnych;

Metoda wyróżnionych zer i jedynek - opis algorytmu syntezy asynchronicznych układów sekwencyjnych, wyznaczanie funkcji wzbudzeń przerzutników stanowiących blok pamięci układu bezpośrednio z siatek przejść i tablicy wzbudzeń przerzutnika *sr*.

Tekst teorii podzielony jest na numerowane strony, co ułatwia orientację. Ilustrowany jest rysunkami pomagającymi w zrozumieniu opisywanych zagadnień. Na dole ekranu zawsze widoczna jest linia statusu z listą klawiszy, którym w danej chwili przypisane są funkcje.

3.2.2. Opcja PRZYKŁADY

Fragment programu reprezentowany przez tę opcję charakteryzuje się cechami programu demonstracyjnego. Stanowi dokładną ilustrację do przedstawionych w opcji *TEORIA* algorytmów. Umożliwia znalezienie związku między teorią a praktyką. Na konkretnych przykładach pokazane są kolejne fazy rozwiązania:

— wyróżnianie w siatkach stanu zer i jedynek w polach odpowiadających stanom niestabilnym,
— zapisanie na tej podstawie wyrażeń w kanonicznej postaci sumy opisujących funkcje sterujące przerzutników dla funkcji stanu układu,
— wyznaczenie na podstawie siatki wyjścia minimalnej postaci sumy dla funkcji wyjścia,
— realizacja wyznaczonych funkcji z zastosowaniem zespołu FPLS.

Podopcja *Przykład 1* prezentuje tylko sposób wyznaczania połączeń w matrycach AND i OR zespołu dla danych funkcji. Proces ukazany jest "krok po kroku". Struktura FPLS widoczna jest cały czas na ekranie. W polu poniżej zmieniają się kolejne realizowane funkcje podane w postaci sumy iloczynów. Zaznaczenie każdego połączenia w matrycy AND widoczne jako "migająca kropka" poprzedzone jest wyróżnieniem czynnika, któremu odpowiada to połączenie. Czynnik ten widoczny jest na ekranie jako "spadająca literka". Po zaznaczeniu połączenia dla każdego ostatniego czynnika w składniku sumy zaznaczane jest połączenie linii, na której zrealizowany został ten iloczyn, z odpowiednią linią poziomą matrycy OR.

Na monitorach kolorowych każde wyrażenie opisujące funkcję wyświetlane jest w innym kolorze. Połączenia w matrycach zaznaczane są w kolorach funkcji, którym odpowiadają. Ułatwia to orientację w dość gęstej sieci linii matryc. Animacja procesu ma na celu uatrakcyjnienie pokazu. Ponadto z klawiatury sterować można szybkością akcji, względnie zatrzymać ją na dowolnie długi czas w celu dokładniejszego prześledzenia zachodzących zmian. Można również zrezygnować z dalszego oglądania prezentacji przez przyciśnięcie klawisza ESC, który pełni identyczną funkcję w całym programie i oznacza rezygnację z rozpoczętej akcji.

Pozostałe podopcje: Przykład 2, Przykład 3 i Przykład 4 przedstawiają cały proces syntezy automatu sekwencyjnego, począwszy od siatek przejść i wyjść. Starano się dane do przykładów dobrać w taki sposób, by obejmowały podstawowe przypadki syntezy.

Przykłady podzielone są na numerowane strony obrazujące kolejne kroki rozwiązania. Pierwsza strona zawiera zawsze treść przykładu, każda następna podzielona jest na trzy obszary (okna), w których wyświetlane są dane wejściowe (siatki), kolejne przekształcenia na nich i funkcje na tej podstawie tworzone oraz komentarz do każdego przekształcenia. Pomiędzy poszczególnymi stronami przykładu można się przemieszczać w przód i w tył, analizując dowolnie długo każdy etap rozwiązania.

3.2.3. Opcja ZADANIA

Opcja ta pozwala użytkownikowi na sprawdzenie swojej wiedzy i umiejętności, a prowadzącemu laboratorium udostępnia narzędzie umożliwiające dodawanie nowych zadań do zestawu już istniejących.

3.2.3.1. Rozwiązywanie zadań

Ten fragment programu charakteryzuje się cechami właściwymi programom ćwiczeniowym. Uczący się rozwiązuje zadania różnych typów i o różnym stopniu trudności, obejmujące zakres materiału uwzględniony w opcjach *TEORIA* i *PRZYKŁADY*. Na każdym etapie rozwiązania sprawdzana jest jego poprawność; rozwiązujący zadanie na bieżąco informowany jest o błędach, które popełnia, udzielane są mu wskazówki, ewentualnie sugeruje mu się uzupełnienie wiadomości w części teoretycznej lub przykładowej.

Dane do zadań przechowywane są w plikach dyskowych o specjalnym formacie. Do programu dołączono kilkanaście takich plików, a system umożliwia tworzenie nowych.

Zadania przeznaczone do samodzielnego rozwiązania najogólniej podzielić można na dwa typy:

— typ 1 - ich rozwiązanie polega na realizacji w matrycach zespołu FPLS funkcji, które w zadaniu są dane;

— typ 2 - synteza automatu Moore'a lub Mealy'ego, dane są siatki przejść i wyjść, należy wyznaczyć funkcje wzbudzeń przerzutników i zrealizować je w zespole FPLS.

Opcja *ZADANIE* udostępnia pięć podopcji: *ZADANIE TYPU 1, ZADANIE TYPU 2 (aut. Moore'a), ZADANIE TYPU 2 (aut. Mealy'ego), ZADANIE ZE ZBIORU i TWORZENIE NOWEGO ZADANIA.*

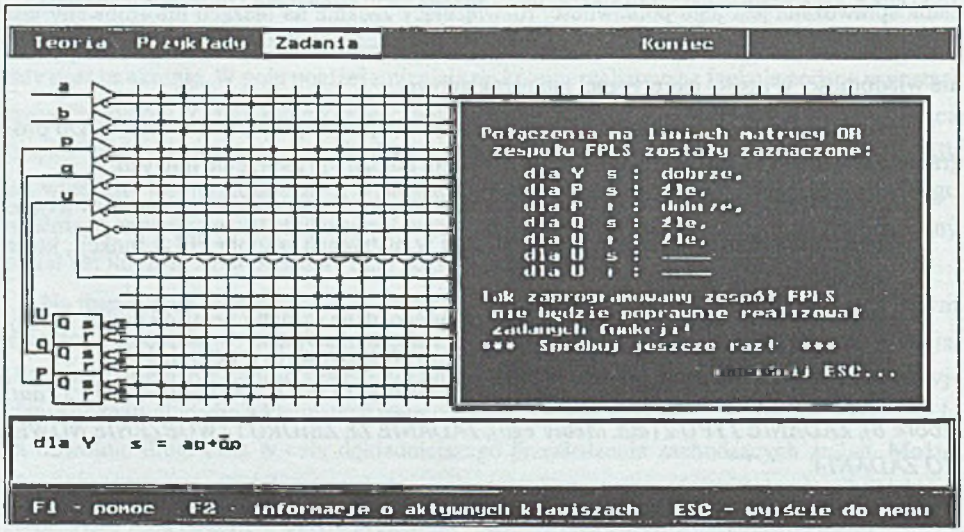
Dla pierwszych trzech podopcji zaimplementowano w programie specjalny mechanizm wyboru zbioru z danymi do zadania. Zbiory te przechowywane są w katalogu podrzędnym w stosunku do katalogu, z którego uruchamiany jest program i tylko tam są poszukiwane. Dla każdego typu zadań zbiory mają różne rozszerzenia nazwy. Po wybraniu którejś z trzech

pierwszych podopcji najpierw szukany jest zbiór o nazwie "staly" z odpowiednim rozszerzeniem. Dopiero gdy nie zostanie on znaleziony, losowany jest jeden z istniejących w danym katalogu zbiorów dla wybranego typu zadania. Mechanizm ten wprowadzono w tym celu, by prowadzący laboratorium, jeśli uzna jakieś zadanie za szczególnie warte zaprezentowania studentom, mógł włączyć je na stałe do wybranej podopcji menu właśnie przez umieszczenie go w zbiorze "staly" z właściwym rozszerzeniem.

Podopcja ZADANIE TYPU 1

Po wybraniu tej jak i każdej innej reprezentującej zadanie podopcji wyświetlana jest treść zadania. Po zapoznaniu się z nią i naciśnięciu dowolnego klawisza na ekranie pojawia się rysunek zespołu FPLS z prostokątnym kursorem widocznym na skrzyżowniu linii matrycy AND. Przesuwać go można po całym obszarze matryc sterując odpowiednio z klawiatury. Wyrażenia opisujące funkcje sterujące elementów pamięci wyświetlane są w oknie poniżej rysunku zespołu. Ze względu na dużą ich liczbę i rozmiary matryc wyświetlana jest zawsze tylko jedna z funkcji; sterowanie z klawiatury umożliwia ich wymianę.

Po zaznaczeniu przez rozwiązującego zadanie połączeń i zaakceptowaniu przez niego postaci matryc jako ostatecznego rozwiązania następuje sprawdzenie jego poprawności. Jeżeli zadanie rozwiązane zostało poprawnie, to wyświetlany jest komunikat o pomyślnym zakończeniu procesu rozwiązania. Jeśli jednak wykryty zostanie błąd, to na ekranie pojawia się okno z raportem (rys. 11), w którym wskazane jest, które z funkcji zrealizowane zostały poprawnie,

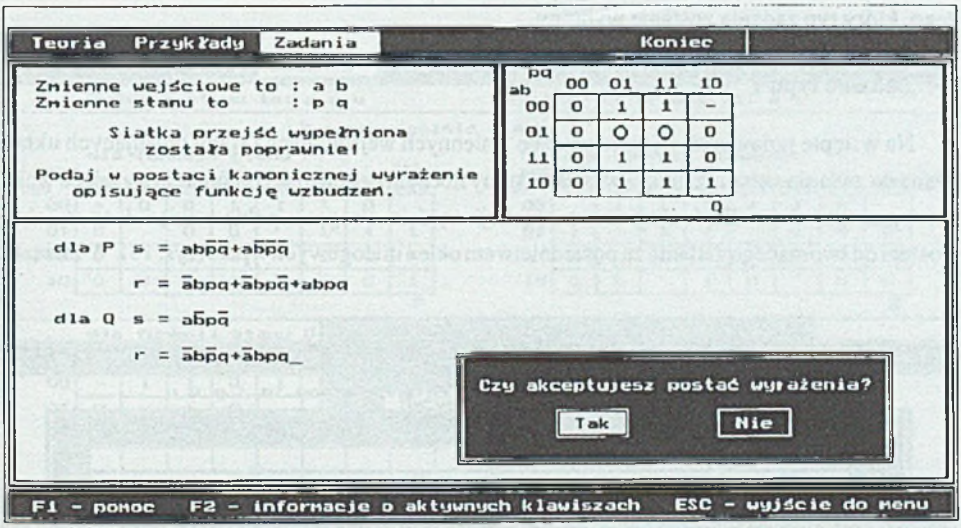


Rys.11. Okno raportu w opcji Zadania/Zadanie typu 1
Fig.11. The report-window in Zadania/Zadanie typu 1 option

a które błędnie. Użytkownik ma możliwość dwukrotnego ponowienia próby wypełniania matryc zespołu. W przypadku gdy zadanie zostało rozwiązane błędnie po raz trzeci, sygnalizowane jest to stosownym komunikatem i prezentowane jest rozwiązanie poprawne. Następuje wyjście z zadania i powrót do menu głównego.

Podopcje *ZADANIE TYPU 2 (aut. Moore'a)* i *ZADANIE TYPU 2 (aut. Mealy'ego)*

Dla zadań tego typu dane wejściowe pobierane ze zbioru dyskowego mają postać siatek stanu i wyjść. Wyświetlane są one na wstępie wraz z treścią zadania. Dla zadań typu 2 ekran podzielony jest na trzy części (rys. 12). W lewym górnym oknie wyświetlane są komunikaty i polecenia dla użytkownika, w prawym górnym - siatki, na których dokonuje się operacji i na podstawie których wyznacza się funkcje wzbudzeń przerzutników, wpisywane jako wyrażenia w kanonicznej postaci sumy w oknie dolnym.



Rys.12. Wygląd ekranu w opcji Zadania/Zadanie typu 2 (aut. Moore'a)
Fig.12. Image of the screen in Zadania/Zadanie typu 2 (aut.Moore'a) option

Poprawność rozwiązania kontrolowana jest po każdym jego kroku. Przez krok rozwiązania rozumie się analizę stanów niestabilnych w siatce bądź podanie na jej podstawie postaci odpowiedniej funkcji. Wyróżnianie wartości w siatkach stanu i wyznaczanie wyrażeń dla funkcji sterujących przerzutników trwa na przemian do czasu, gdy wszystkie funkcje zostaną poprawnie podane lub ćwiczący popełni czwarty błąd. W ostatnim przypadku rozwiązywanie zadania zostaje przerwane. Jeżeli natomiast wszystkie funkcje dla siatek przejść i wyjścia podane zostaną poprawnie, użytkownik ma prawo rozpocząć drugi etap rozwiązania, czyli wypełnianie matryc zespołu FPLS. Przebiega on tak, jak dla podopcji *ZADANIE TYPU 1*.

Podopcja *ZADANIE ZE ZBIORU*

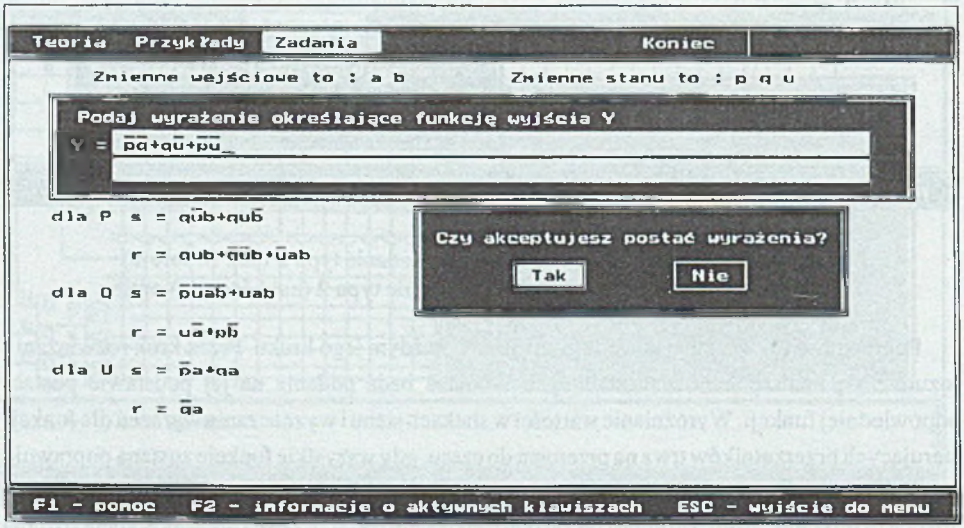
Podopcja ta wprowadzona została w celu umożliwienia korzystającemu z programu wyboru dowolnego zbioru z danymi do zadania. Po jej wybraniu pojawia się na ekranie prośba o wybór typu zadania do rozwiązania wraz z krótką charakterystyką tych typów. Po dokonaniu wyboru wyświetlana jest lista z nazwami dostępnych dla tego typu zbiorów i pole dialogowe wpisu, za pośrednictwem którego można podać nazwę wybranego zbioru. Rozwiązanie zadania przebiega w zależności od typu tak, jak to przedstawiono dla trzech pierwszych podopcji.

3.2.3.2. Tworzenie nowych zbiorów z zadaniami

Podopcja *TWORZENIE NOWEGO ZADANIA* przeznaczona jest głównie dla prowadzącego laboratorium. Umożliwia utworzenie nowych zbiorów przechowujących dane do zadań, które będą mogły być później rozwiązywane. Procesy tworzenia zadań różnią się w zależności od tego, który typ zadania zostanie wybrany.

Zadanie typu 1

Na wstępie pojawia się pytanie o liczbę zmiennych wejściowych i stanu opisujących układ. Dane do zadania są wyrażeniami w postaci sumy iloczynów opisującymi funkcje sterujące wejść s i r przerzutników dla funkcji stanu i funkcji wyjścia. Program pobiera więc dane w takiej postaci od tworzącego zadanie za pośrednictwem okien dialogowych wpisu (rys. 13). W związku

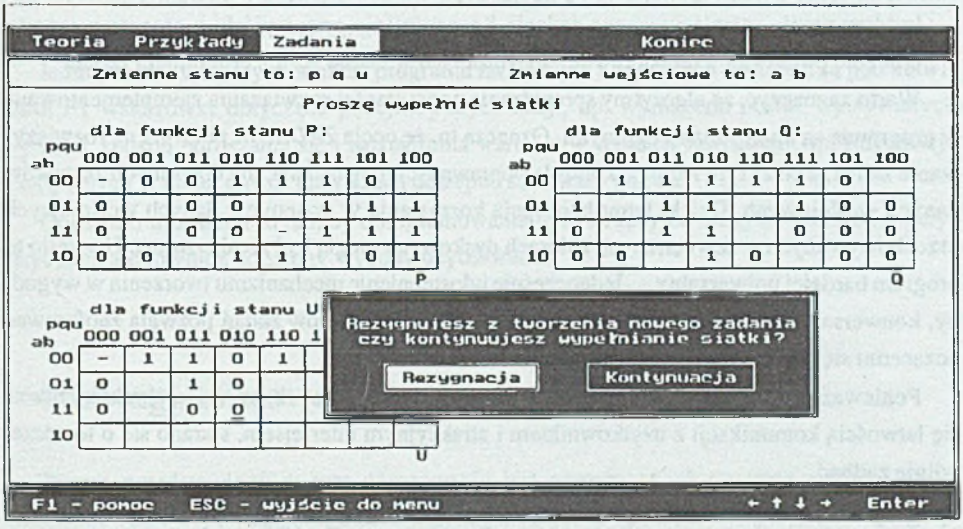


Rys.13. Tworzenie zadania typu 1 - okienka wpisu i akceptacji
Fig.13. Creating of 1-st type exercise - input-line and acceptation dialog-box

z ograniczeniami nałożonymi przez wielkość i rozdzielczość ekranu konieczne było wprowadzenie pewnych ograniczeń na wymiary prezentowanego zespołu. Hipotetyczna struktura występująca w programie ma 32 linie w matrycy AND i 8 linii w matrycy OR. Wystarcza to jednak w pełni na zrealizowanie układu sekwencyjnego o jednym wyjściu, opisanego dwoma zmiennymi wejściowymi i trzema zmiennymi stanu, jeśli liczba różnych iloczynów w wyrażeniach opisujących funkcje nie przekracza 32.

Zadanie typu 2

Dane do zadania tego typu mają postać siatek przejść i siatki wyjścia opisujących układ sekwencyjny. Użytkownik wybiera rodzaj automatu (Moore'a lub Mealy'ego) i liczbę zmiennych opisujących układ. W zależności od tych wartości wyświetlane są siatki o odpowiedniej liczbie pól, które należy wypełnić (rys.14).



Rys.14. Tworzenie zadania typu 2
Fig.14. Creating of 2-nd type exercise

Po zatwierdzeniu postaci ostatniej z siatek (lub podaniu wszystkich wyrażeń dla zadania typu 1) i podaniu nazwy nie istniejącego pliku lub wyrażeniu zgody na nadpisanie istniejącego pliku dane są zapisywane, a pomyślne zakończenie tego procesu sygnalizowane jest komunikatem.

3.3. Podsumowanie

Z każdej z trzech części programu reprezentowanych przez opcje *TEORIA, PRZYKŁADY* i *ZADANIA* można korzystać oddzielnie, w zależności od tego, jak uczący się sam ocenia swoją wiedzę na temat prezentowanych zagadnień. Aby skorzystać z części ćwiczeniowej, nie jest się zobligowanym do zapoznania się z przedstawioną w programie teorią czy przykładami. Dzięki temu program może się okazać przydatny np. dla studentów przygotowujących się do kolokwium i chcących upewnić się, czy rzeczywiście prawidłowo zrozumieli i opanowali materiał programowy. Ponadto przez przez taki a nie inny sposób prezentacji zagadnień starano się osiągnąć dwa podstawowe cele dydaktyczne:

— przekazać w jak najbardziej przystępnej formie pewien zakres wiedzy za pośrednictwem bogato skomentowanej i zaopatrzonej w barwną grafikę i animację części teoretycznej oraz przykładów,

— zmobilizować uczącego się do aktywnej pracy przez kontrolę poprawności rozwiązania na każdym etapie rozwiązywanego zadania i generowanie przy wyjściu z programu raportu o liczbie poprawnie i niepoprawnie rozwiązanych w czasie sesji z programem zadań.

Warto zaznaczyć, że algorytmy sprawdzania poprawności rozwiązania zaimplementowane w programie są algorytmami ogólnymi. Oznacza to, że opcja *ZADANIA* pozwala na rozwiązywanie zadań, a co za tym idzie na kontrolę poprawności rozwiązania, dla dowolnych zestawów danych wejściowych. Dzięki temu koncepcja korzystania w systemie z danych wejściowych przechowywanych w zewnętrznych zbiorach dyskowych mogła być zrealizowana. Uczyniło to program bardziej uniwersalny. Jednocześnie udostępnienie mechanizmu tworzenia w wygodny, konwersacyjny sposób zbiorów z danymi dla trzech rodzajów zadań pozwala zaoferować uczącemu się wiele zadań o różnym stopniu trudności.

Ponieważ program edukacyjny oprócz walorów dydaktycznych powinien chrakteryzować się łatwością komunikacji z użytkownikiem i atrakcyjnym interfejsem, starano się o to szczególnie zadbać.

Program napisano dla trybu graficznego karty video. Pozwoliło to przede wszystkim usunąć jeden z problemów, z którym od lat borykają się z lepszym lub gorszym skutkiem twórcy oprogramowania. Uniezależniono się mianowicie od standardu polskich znaków w trybie tekstowym karty. W związku z tym teksty wyświetlane mogą być w programie zgodnie z polską pisownią, z uwzględnieniem wszystkich znaków alfabetu polskiego.

Oprogramowanie wyposażono w prosty w obsłudze i jednocześnie wygodny graficzny interfejs użytkownika, przypominający, jeśli chodzi o zasady komunikacji, pośrednictwo popularnych systemów okienkowych. Rozplanowanie na ekranie elementów graficznych pośredniczących w komunkacji nie odbiega zasadniczo od standardu wprowadzonego przez firmę Microsoft [7]. Program oferuje więc menu z rozwijanymi podmenu typu "pop-up", wiersz

statusu, w którym zawsze wyświetlana jest lista aktywnych klawiszy i obszar roboczy ekranu przeznaczony do wyświetlania okien w czasie pracy programu.

Powstał również system okien dialogowych: wyboru, wpisu i komunikatu, ułatwiający komunikację. Przykłady okienek dwóch pierwszych typów widoczne są na rys. 13. Szczególnie często używane są w programie okna wyboru. Między innymi po każdym przyciśnięciu klawisza ENTER lub ESC korzysta się z takiego okna wyświetlając prośbę o potwierdzenie akceptacji lub rezygnacji z akcji. Zabezpieczenie takie wprowadzono dla wygody pracującego z programem, gdyż np. przerwanie żmudnego procesu tworzenia lub rozwiązywania zadania przez nieumyślne wybranie ESC lub ENTER zmuszałoby operatora do powtarzania praco- i czasochłonnych czynności od początku.

Okna komunikatów służą głównie do sygnalizacji pomyślnego lub niepomyślnego zakończenia akcji oraz ostrzeżeń. Poza tym program charakteryzuje się wszechstronnym zabezpieczeniem przed wprowadzaniem danych w postaci niezgodnej z wymaganiami. Niedozwolone i nielogiczne z punktu widzenia procesów związanych z pobieraniem danych akcje podejmowane przez użytkownika również powodują wyświetlenie stosownych komunikatów.

Jednocześnie w każdym punkcie programu związanym z zadaniami dostępne są pod klawiszem F1 wskazówki dotyczące podejmowanych akcji, np. wymagana postać wpisywanych wyrażeń, sposób poruszania się i zaznaczania wartości w siatkach Karnaugha itp. Informacje o aktywnych w danej chwili klawiszach dostępne są zawsze pod F2.

Dołożono wszelkich starań, by oprogramowanie było atrakcyjne nie tylko od strony merytorycznej, ale również od strony wygody użytkowania.

4. Uwagi końcowe

System przedstawiony w tym opracowaniu jest pierwszą i jak na razie jedyną wersją oprogramowania wspomagającego nauczanie zagadnień zespołów FPLS i w przyszłości mógłby zostać uzupełniony nowymi możliwościami.

W ramach poszerzenia zakresu prezentowanego materiału można by w programie zaimplementować zmodyfikowaną "metodę wyróżnionych zer i jedynek", uwzględniającą tworzenie grup w siatkach przejść, co w konsekwencji pozwala na uzyskanie skróconej postaci wyrażeń opisujących funkcje. Z punktu widzenia twórcy zadań do programu atrakcyjne byłoby przypuszczalnie dodanie opcji edycji istniejących już zbiorów z danymi. Jeżeli natomiast chodzi o komunikację program - użytkownik, to możliwość posługiwania się myszą na pewno by ją poprawiła.

Poza tym obserwuje się tendencje do przechodzenia ze środowiska DOS w środowisko systemu Windows, które zdaje się być wyzwaniem dla coraz szerszego grona programistów. Przeniesienie oprogramowania edukacyjnego do Windows może nadać mu całkiem nową jakość. Sterowanie przepływem komunikatów i programowanie zorientowane obiektowo dają bowiem twórcy aplikacji nowe możliwości, które wydają się być nie do pogardzenia.

Informacje na temat udostępnienia programu uzyskać można pod adresem internetowym: *atomasz@star.iinf.gliwice.edu.pl.*

LITERATURA

[1] Górna D.: - *Opracować algorytm i oprogramowanie umożliwiające syntezę asynchronicznych statycznych układów sekwencyjnych realizowanych z zastosowaniem programowanych zespołów logicznych PLA.* Praca dyplomowa magisterska. Instytut Informatyki Politechniki Śląskiej, Gliwice 1989.

[2] Kamionka-Mikuła H.: *Analiza przypadków nieszkodliwości hazardu w automatach sekwencyjnych.* Praca doktorska. Politechnika Śląska, Gliwice 1974.

[3] Małysiak H., Pochopień B. (red.): *Układy cyfrowe - zadania.* Skrypt Politechniki Śląskiej, nr 1296, Gliwice 1992.

[4] Pieńkos J., Turczyński J.: *Układy scalone TTL w systemach cyfrowych.* WKŁ, Warszawa 1986.

[5] Siwiński J.: *Układy przełączające w automatyce.* WNT, Warszawa 1980.

[6] Tomaszewska A.: - *Opracowanie oprogramowania wspomagającego nauczanie w zakresie zagadnień zespołów FPLS.* Praca dyplomowa magisterska. Instytut Informatyki Politechniki Śląskiej, Gliwice 1993.

[7] *Common User Acces Advanced Interface Designed Guide* - IBM System Application Architecture; doc. nr SY0328-300-R00-1089, First Edition, June 1989.

[8] Prace dyplomowe wykonywane w latach 1979-91 pod kierunkiem H. Kamionki-Mikuły w Instytucie Informatyki Politechniki Śląskiej w Gliwicach.

Recenzent: Dr inż. Jerzy Mikulski

Wpłynęło do Redakcji 16 listopada 1994 r.

Abstract

Field Programmable Logic Sequencers belong to programmable logic arrays - units that serve for logic function realization. However the structure of FPLS unit, as shown in fig.1, differs from other PLA. FPLS units enable to implement not only the combinatorial Boolean functions but, due to the feedback using the state register, the sequential functions as well.

State functions (eq. 2.2.2) that are characteristic for the asynchronous sequential switching circuits (see the diagram in fig.2), are just the sequential functions. Therefore using Field Programmable Logic Sequencers let designers realize such circuits in a simple way. Computer-assisted instruction system presented in this paper concerns exactly these problems.

The range of subject matter comprises a process of synthesis of the asynchronous sequential switching circuts, what is thoroughly described in chapter 2. This process starts from the analysis of unsettled states in state K-maps. On the ground of these K-maps and rs excitation-table set- and reset-functions of *rs* flip-flop gates can be calculated (section 2.3.1). After calculating of set- and reset-functions for the output function from output K-map (section 2.3.2) all these Boolean expressions should be implemented in the programmable arrays of FPLS unit (fig.9). This ends the synthesis process. All these problems are included in program presented in this article.

The system is divided into three parts represented by three menus: "Teoria", "Przykłady" and "Zadania". Each of these parts plays different role in the instruction process, what is described in chapter 3 .

The first part (section 3.2.1) helps the user to learn basic definitions and algorithms concerning synthesis of asynchronous sequential switching circuits and FPLS units. The second part (section 3.2.2) gives the examples which let a student find the connections between theory and practice. The third part (section 3.2.3) which is the most important offers a user a rich set of exercises, which may be also extended. By solving the problems given in these exercises students may improve their skills and check their real knowledge. This part was designed for training. The universal algorithms which let check the solution correctness for any sets of input data, were implemented.

Section 3.4. describes the facilities and improvements concerning communication between user and program implemented in the presented computer-assisted instruction software. The examples of user interface were given in fig.10 - fig.14.

The suggestions for system developing were discussed in chapter 4.