Układy programowalne, część 6

Kontynuujemy prezentację przykładowych opisów w języku CUPL, w tej części skupiając się na układach kombinacyjnych. Kody źródłowe prezentowanych projektów wraz z plikami symulacyjnymi publikujemy na CD-EP8/2004B. Gorąco zachęcamy do samodzielnych prób, do których można wykorzystać zestaw ewaluacyjny AVT-559.

Jak już wcześniej wspomniano, za pomocą języka CUPL można opisywać projektowane sprzętowe bloki funkcjonalne na wiele sposobów. Najbardziej oczywistym i przy tym najmniej wygodnym są równania boole'owskie, odpowiadające w nomenklaturze mikroprocesorowej pisaniu programów w asemblerze. Pokażemy teraz kilka przykładów rozwiązania prostych, aczkolwiek często napotykanych w praktyce, problemów za pomocą różnych sposobów opisu.

Dekoder adresowy

Zaprojektujemy dekoder adresowy z trzema wyjściami (RAM_SEL, IO_SEL, ROM_SEL), które uaktywniają

RAM_SEL	ROM_SEL	IO_SEL
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	11	11
12	12	12
13	13	13
14	14	14
15	15	15
16	16	16
17	17	17
18	18	18
19	19	19
20	20	20
21	21	21
22	22	22
23	23	23
24	24	24
25	25	25
26	26	26
27	27	27
28	28	28
29	29	29
30	30	30
31	31	31

Rys. 28

(poziom aktywny tych sygnałów to „1") bloki peryferyjne systemu cyfrowego w zależności do stanu wejść *adresowych Adr4...Adr0* (32 różne adresy). Mapę przykładowego obszaru adresowego pokazano na **rys. 28** (zaciemnione pola wskazują peryferia aktywne pod danym adresem).

Zminimalizowane równania logiczne zapewniające realizację przez układ PLD funkcji zgodnie z podaną specyfikacją dla wyjść IO_SEL i RAM_SEL przedstawiono na **list. 4**. Skonstruowanie tych równań, jakkolwiek możliwe, jest jednak dość kłopotliwe i znacznie utrudnia wprowadzenie do projektu ewentualnych zmian jak np. przesunięcia lokalizacji peryferiów w przestrzeni adresowej. Znacznie lepszym i wygodniejszym wyjściem jest zapisanie projektu w sposób pokazany na **list. 5**. W opisie tym zastosowano operator przypisania (:), za pomocą którego wcześniej zadeklarowanym wektorom są przypi-

Kierunek linii I/O

Projektant przygotowując opis HDL za pomocą CUPL-a nie musi (nie ma jak) zadeklarować kierunków sygnałów przypisanych do wyprowadzeń (wejścia/wyjścia/wejścia-wyjścia). Kompilator ustala kierunki samoczynnie na podstawie opisu i w odniesieniu do fizycznych możliwości docelowego układu PLD.

Inne możliwości stosowania operatora przypisania

Operator przypisania można wykorzystać do skrócenia zapisu równań logicznych dla operatorów działań: &, # i $.

Przykładowo zapisy:

```
[A3,A2,A1,A0]:&
[B3,B2,B1,B0]:#
[A,B,C,D]:$
```

odpowiadają równaniom:

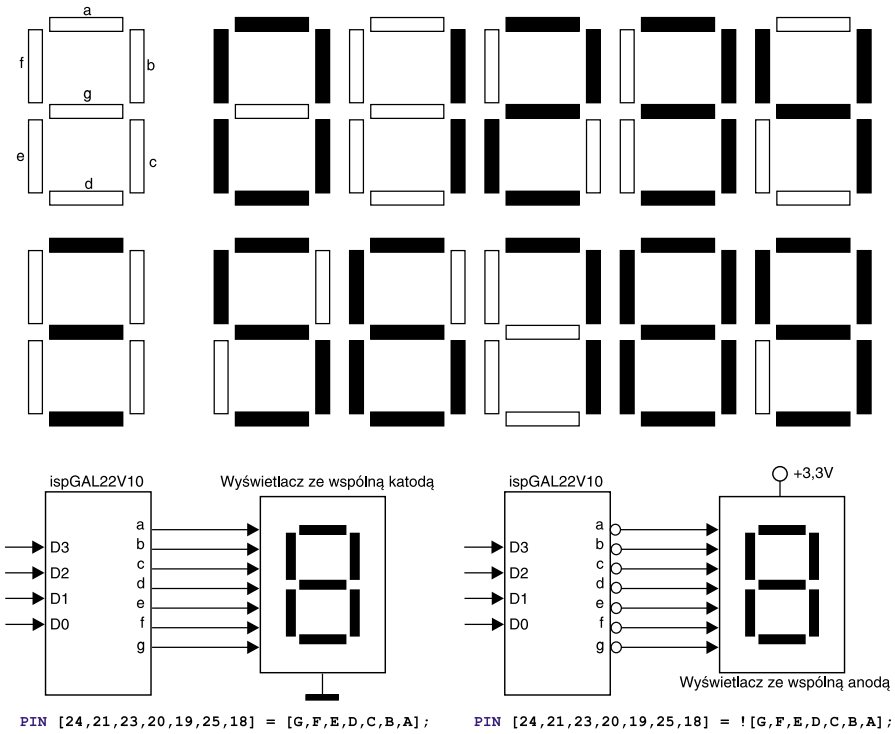
```
A3 & A2 & A1 & A0
B3 # B2 # B1 # B0
A $ B $ C $ D
```

List. 4. Równania boole'owskie funkcji logicznych dla wyjść IO_SEL i RAM_SEL (funkcje zgodnie z rys. 28)

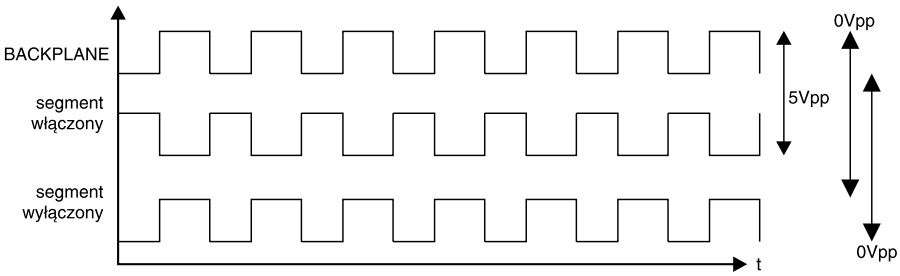
```
IO_SEL = Adr1 & Adr2 & !Adr3 & !Adr4
 # !Adr1 & !Adr2 & Adr3 & !Adr4
 # !Adr0 & Adr1 & !Adr2 & Adr3 & !Adr4
 # !Adr0 & !Adr1 & Adr2 & !Adr3 & Adr4
 # Adr0 & Adr1 & Adr2 & Adr3 & Adr4

RAM_SEL = Adr0 & Adr1 & Adr3 & !Adr4
 # !Adr1 & Adr2 & Adr3 & !Adr4
 # !Adr0 & Adr1 & Adr2 & Adr3 & !Adr4
 # !Adr2 & Adr4
 # Adr0 & Adr2 & !Adr3 & Adr4
 # !Adr0 & Adr1 & Adr2 & !Adr3 & Adr4
```

List. 5. Listing projektu dekodera adresów z pięcioma wejściami i trzema wyjściami (funkcje zgodnie z rys. 28)

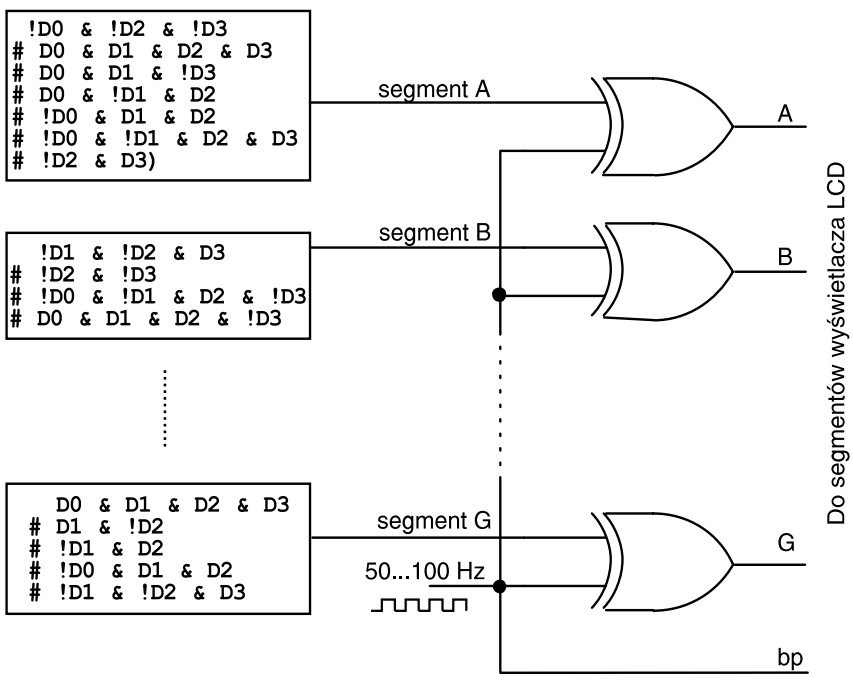
```
Name            dekoder;
Partno          brak;
Revision        brak;
Date            20/05/04;
Designer        PZb;
Company         EP;
Location        brak;
Assembly        brak;
Device          g22v10lcc;

/* Adres ustala sie za pomoca nastawnika
/* SW1(Adr0...Adr3)
/*     az jumpera JP1 (Adr4)            */

/***** Wejscia *****/
PIN [7,9..11] = [Adr3..0];
PIN 4 = Adr4; /* Jumper JP1 */

/***** Wyjscia *****/
PIN [26,23,17] = [RAM_SEL,IO_SEL,ROM_SEL];

/***** Deklaracje pomocnicze *****/
field ADRES = [Adr4..0];
serport_tx = ADRES:['d'6..'d'10];
serport_rx = ADRES:'d'20 # ADRES:'d'31;
ram_rd = ADRES:['d'11..'d'19];
ram_wr = ADRES:['d'21..'d'27];

/***** Opis HDL *****/
RAM_SEL = ram_rd # ram_wr;
IO_SEL = serport_tx # serport_rx;
ROM_SEL = ADRES:['d'0..'d'5] # ADRES:
['d'28..'d'30];
```



Rys. 29



Rys. 30



Rys. 31

List. 6. Projekt dekodera wyświetlacza 7-segmentowego opisanego równaniami logicznymi

```
Name            dek_wys;
Partno          U1;
Revision        01;
Date            20/05/04;
Designer        PZb;
Company         EP;
Location        brak;
Assembly        brak;
Device          g22v10lcc;

/*Stany na wejsciach D3...D0 ustala sie za */
/*    pomoca nastawnika SW1                 */
/*****************************/
/*          a                */
/*        -----              */
/*       |     |             */
/*     f|     |b             */
/*      |  g  |              */
/*       -----               */
/*      |     |              */
/*     e|     |c             */
/*      |     |              */
/*       -----               */
/*          d                */
/*                           */
/*****************************/

/***** Wejscia *****/
PIN [7,9..11] = [D3..0];

/***** Wyjscia *****/
PIN [24,21,23,20,19,25,18] = [G,F,E,D,C,B,A];

/***** Deklaracje pomocnicze *****/
Field dana = [D3..0];
Field segment = [A,B,C,D,E,F,G];

/***** Opis HDL *****/
A = !D0 & !D2 & !D3
  # D0 & D1 & D2 & D3
  # D0 & D1 & !D3
  # D0 & !D1 & D2
  # !D0 & D1 & D2
  # !D0 & !D1 & D2 & D3
  # !D2 & D3;

B =  !D1 & !D2 & D3
  # !D2 & !D3
  # !D0 & !D1 & D2 & !D3
  # D0 & D1 & D2 & !D3;

C = !D1 & !D2 & D3
  # !D0 & D1 & D2 & !D3
  # D0 & D1 & !D3
  # !D1 & !D3;

D = !D0 & !D2 & !D3
  # D0 & D1 & D2 & D3
  # D0 & D1 & !D2
  # D0 & !D1 & D2
  # !D0 & D1 & D2
  # !D1 & !D2 & D3
  # !D0 & D1 & !D2 & D3
  # !D0 & !D1 & D2 & D3;

E = !D0 & !D2 & !D3
  # !D0 & !D1 & !D2 & D3
  # !D0 & D1 & D2 & !D3;

F = !D0 & !D1 & !D3
  # !D1 & !D2 & D3
  # D0 & !D1 & D2 & !D3
  # !D0 & D1 & D2 & !D3;

G = D0 & D1 & D2 & D3
  # D1 & !D2
  # !D1 & D2
  # !D0 & D1 & D2
  # !D1 & !D2 & D3;
```

sywane oczekiwane wartości lub ich przedziały, jak np.: `serport_tx = ADRES:['d'6..'d'10]`. Przypisanie może mieć także postać równania logicznego, jak np.: `serport_rx = ADRES:'d'20 # ADRES:'d'31`. Tak zapisane równania kompilator sam rozwinie do postaci „czystych" równań logicznych, znacznie ułatwiając projektantowi diagnostykę projektu i jego ewentualną modyfikację.

Dekoder-sterownik wyświetlacza 7-segmentowego

W kolejnym przykładzie przedstawimy trzy możliwe sposoby opisu dekodera 7-segmentowego współ-

List. 7. Projekt dekodera wyświetlacza 7-segmentowego opisanego równaniami logicznymi o postaci łatwej do weryfikacji przez projektanta

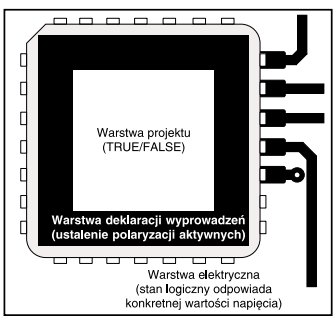
```
Name            dek_wys;
Partno          U1;
Revision        01;
Date            20/05/04;
Designer        PZb;
Company         EP;
Location        brak;
Assembly        brak;
Device          g22v10lcc;

/* Stany na wejsciach D3...D0 ustala sie za  */
/*   pomoca nastawnika SW1                   */
/*************************/
/*          a            */
/*       -----           */
/*      |     |          */
/*     f|     |b         */
/*      |  g  |          */
/*       -----           */
/*      |     |          */
/*     e|     |c         */
/*      |     |          */
/*       -----           */
/*         d             */
/*                       */
/*************************/

/***** Wejscia *****/
PIN [7,9..11] = [D3..0];

/***** Wyjscia *****/
PIN [24,21,23,20,19,25,18] = [G,F,E,D,C,B,A];

/***** Deklaracje pomocnicze *****/
Field dana = [D3..0];
Field segment = [A,B,C,D,E,F,G];

$define ON  'b'1        /* Segment swieci     */
$define OFF 'b'0        /* Segment nie swieci */

/***** Opis HDL *****/
/*              a    b    c    d    e    f    g  */
segment   =
/* 0 */       [ ON,  ON,  ON,  ON,  ON,  ON, OFF] & dana:0
/* 1 */    #  [OFF,  ON,  ON, OFF, OFF, OFF, OFF] & dana:1
/* 2 */    #  [ ON,  ON, OFF,  ON,  ON, OFF,  ON] & dana:2
/* 3 */    #  [ ON,  ON,  ON,  ON, OFF, OFF,  ON] & dana:3
/* 4 */    #  [OFF,  ON,  ON, OFF, OFF,  ON,  ON] & dana:4
/* 5 */    #  [ ON, OFF,  ON,  ON, OFF,  ON,  ON] & dana:5
/* 6 */    #  [ ON, OFF,  ON,  ON,  ON,  ON,  ON] & dana:6
/* 7 */    #  [ ON,  ON,  ON, OFF, OFF, OFF, OFF] & dana:7
/* 8 */    #  [ ON,  ON,  ON,  ON,  ON,  ON,  ON] & dana:8
/* 9 */    #  [ ON,  ON,  ON,  ON, OFF,  ON,  ON] & dana:9
/* A */    #  [ ON, OFF, OFF,  ON, OFF, OFF,  ON] & dana:A
/* B */    #  [ ON, OFF, OFF,  ON, OFF, OFF,  ON] & dana:B
/* C */    #  [ ON, OFF, OFF,  ON, OFF, OFF,  ON] & dana:C
/* D */    #  [ ON, OFF, OFF,  ON, OFF, OFF,  ON] & dana:D
/* E */    #  [ ON, OFF, OFF,  ON, OFF, OFF,  ON] & dana:E
/* F */    #  [ ON, OFF, OFF,  ON, OFF, OFF,  ON] & dana:F;
}
```

List. 8. Projekt dekodera wyświetlacza 7-segmentowego opisanego równaniami za pomocą tablicy prawdy

```
Name            dek_wys;
Partno          U1;
Revision        01;
Date            20/05/04;
Designer        PZb;
Company         EP;
Location        brak;
Assembly        brak;
Device          g22v10lcc;

/* Stany na wejsciach D3...D0 ustala sie za  */
/*   pomoca nastawnika SW1                   */
/*************************/
/*          a            */
/*       -----           */
/*      |     |          */
/*     f|     |b         */
/*      |  g  |          */
/*       -----           */
/*      |     |          */
/*     e|     |c         */
/*      |     |          */
/*       -----           */
/*         d             */
/*                       */
/*************************/

/***** Wejscia *****/
PIN [7,9..11] = [D3..0];

/***** Wyjscia *****/
PIN [24,21,23,20,19,25,18] = [G,F,E,D,C,B,A];

/***** Deklaracje pomocnicze *****/
Field Dana = [D3..0];
Field Segment = [A,B,C,D,E,F,G];

/***** Opis HDL *****/
Table Dana => Segment {

/*  Wejscia      Wyjscia segmentowe             */
/*  ------------------------------------       */
/*  AAAA                                        */
/*  3210          ABCDEFG                       */
'b'0000   =>  'b'1111110; /* 0                 */
'b'0001   =>  'b'0110000; /* 1                 */
'b'0010   =>  'b'1101101; /* 2                 */
'b'0011   =>  'b'1111001; /* 3                 */
'b'0100   =>  'b'0110011; /* 4                 */
'b'0101   =>  'b'1011011; /* 5                 */
'b'0110   =>  'b'1011111; /* 6                 */
'b'0111   =>  'b'1110000; /* 7                 */
'b'1000   =>  'b'1111111; /* 8                 */
'b'1001   =>  'b'1111011; /* 9                 */
'b'1010   =>  'b'1001001; /* blad              */
'b'1011   =>  'b'1001001; /* blad              */
'b'1100   =>  'b'1001001; /* blad              */
'b'1101   =>  'b'1001001; /* blad              */
'b'1110   =>  'b'1001001; /* blad              */
'b'1111   =>  'b'1001001; /* blad              */
}
```

Niuanse negacji

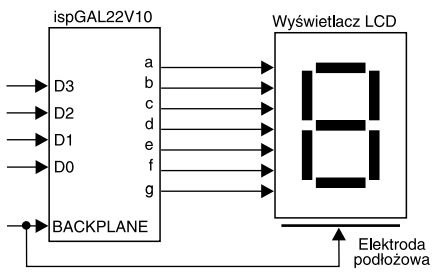
Twórcy CUPL-a przyjęli, że podczas przygotowywania opisu sprzętu projektant rozważa wyłącznie wartości TRUE/FALSE, natomiast o polaryzacji (aktywne „0”/aktywne „1”) sygnału decyduje podczas deklarowania wyprowadzeń.



pracującego z wyświetlaczem LED o wspólnej katodzie. Na **rys. 29** pokazano sposób wyświetlania znaków z zakresu 0...9 oraz znaków o kodach powyżej 9 (palą się wyłącznie poziome segmenty wyświetlacza).

Czytelnicy o największym zacięciu do posługiwania się mapami Karnaugh mogą spróbować zweryfikować poprawność równań logicznych, za pomocą których opisano dekoder w przypadku pokazanym na **list. 6**. Te same równania można zapisać w wygodniejszej postaci (**list. 7**), a ich przekształceniem do postaci pokazanej na list. 6 zajmie się kompilator. Jak można zauważyć, za pomocą deklaracji `define` stałych ON i OFF przypisano wartości bitów (odpowiednio) „1” i „0”, co pozwala posługiwać się w dalszej części opisu czytelnymi nazwami. Taki sposób opisania działania dekodera ułatwia diagnostykę projektu oraz wprowadzania do niego zmian.

Kolejnym możliwym sposobem opisania dekodera jest zawarcie zależności pomiędzy stanami na jego wejściach i wyjściach w tablicy prawdy (**list. 8**). Poszczególnym wartościom wektora wejściowego `Dana` przypisywane są odpowiednie wartości wyjściowe (wektor `Segment`), a całość jest ulokowana w tablicy zaczynającej się od słowa kluczowego `Table`. Obydwa wektory zdefiniowano w polu deklaracji pomocniczych za pomocą słów kluczowych `Field`.



Rys. 32

List. 9. Jeden z możliwych sposobów dodania do funkcji sterującej segmentem A wyświetlacza inwertera sterowanego sygnałem BACKPLANE

```
A = (!D0 & !D2 & !D3
  # D0 & D1 & D2 & D3
  # D0 & D1 & !D3
  # D0 & !D1 & D2
  # !D0 & D1 & D2
  # !D0 & !D1 & D2 & D3
  # !D2 & D3)
  $ BACKPLANE;
```

W przypadku, gdy zaprojektowany sterownik będzie współpracował z wyświetlaczem LED o wspólnej anodzie wystarczy zmienić aktywny stan (z wysokiego na niski) na wyjściach dekodera. Najprostszym sposobem jest zastąpienie linii `PIN [24,21,23,20,19,25,18] = [G,F,E,D,C,B,A];` linią `PIN [24,21,23,20,19,25,18] = ![G,F,E,D,C,B,A];` (w której linie wyjściowe portów zostały zanegowane).

Prezentowany dekoder można łatwo dostosować do sterowania 7-segmentowego wyświetlacza LCD. W tym celu wszystkie wyjścia zasilające segmenty wyświetlacza powinny zostać wyposażone w sterowane inwertery (wykonane np. na bramkach ExOR), które dostarczą do segmentów „świecących" sygnał w przeciwfazie w stosunku do sygnału zasilającego podłoże (*backplane*) wyświetlacza, jak to pokazano na **rys**. **30**. Sterowane inwertery najprościej można uzyskać w CUPL-u XOR-ując funkcje tworzące sygnały sterujące segmentami z sygnałem BACKPLANE, np. w taki sposób jak to pokazano na **list**. **9** (pokazano przykład tylko dla jednego wyjścia).

List. 10. Projekt multipleksera 4-wejściowego

```
Name            mux;
Partno          brak;
Revision        brak;
Date            20/05/04;
Designer        PZb;
Company         EP;
Location        brak;
Assembly        brak;
Device          g22v10lcc;

/* Nastawnik SW1 sluzy do zmiany stanow  */
/*      na wejsciach X3...X0             */
/* Jumpery Sw1 i Sw2 spelniaja role      */
/*      elementow adresujacych           */
/*   aktywne wejscie multipleksera 4x1   */

/***** Wejscia *****/
PIN [7,9..11] = [X0,X1,X2,X3];
PIN [4,6] = [SEL1..0];

/***** Wyjscia *****/
PIN [25] = Y;

/***** Deklaracje pomocnicze *****/
Field SELEKTOR = [SEL1..0];

/***** Opis HDL *****/
Y = (X0 & SELEKTOR:0)
 # (X1 & SELEKTOR:1)
 # (X2 & SELEKTOR:2)
 # (X3 & SELEKTOR:3);
```

Schemat blokowy ilustrujący sposób tworzenia sygnałów sterujących segmentami wyświetlacza LCD pokazano na **rys**. **31**. Na **rys**. **32** przedstawiono sposób dołączenia wyświetlacza do dekodera zaimplementowanego w układzie PLD.

Przedstawiony mechanizm tworzenia tablic prawdy w CUPL-u pozwala na łatwą i wygodną implementacje w układach PLD najróżniejszych tablic przekodowań (transkoderów), często określanych mianem *look-up table*.

Multiplekser

Podobnym do dekodera-sterownika wyświetlacza 7-segmentowego przykładem projektu układu kombinacyjnego jest multiplekser. W artykule pokażemy implementację pojedynczego multipleksera 4-wejściowego.

Najbardziej oczywistym sposobem opisu jest równanie przypisujące wyjściu multipleksera stan występujący na zaadresowanym wejściu. Takie równanie może mieć postać jak poniżej:

```
Y = !SEL0 & !SEL1 & X0
 # SEL0 & !SEL1 & X1
 # !SEL0 & SEL1 & X2
 # SEL0 & SEL1 & X3
```

Podobnie, jak miało to miejsce we wcześniejszych przykładach, taki sposób opisu, jakkolwiek skuteczny, nie jest wygodny. Zdecydowanie lepiej sprawdza się w praktyce (ze względu na wygodę, formalnie obydwa zapisy są praktycznie równoważne) opis pokazany na **list**. **10**.

Jak widać, liczba oferowanych przez CUPL-a sposobów opisu układów kombinacyjnych nie jest duża, ale w zupełności wystarczy do realizacji każdego zadania inżynierskiego, dając przy tym możliwości wybrania przez projektanta sposobu najbardziej mu odpowiadającego.

Za miesiąc przedstawimy kilka przykładów układów synchronicznych, w kolejnych zajmiemy się prezentacją narzędzi.

Piotr Zbysiński, EP
piotr.zbysinski@ep.com.pl