Układy PLD, część 4

Symulator CSIM

W ostatnim artykule poświęconym kompilatorowi CUPL przedstawiamy możliwości symulacji i testowania poprawności pracy projektów realizowanych za pomocą CUPL'a. Dzięki wbudowanemu w kompilator symulatorowi CSIM można „na sucho" (symulacja wirtualna) zweryfikować pracę realizowanego przez nas układu bez konieczności fizycznego programowania struktury półprzewodnikowej. Jest to bardzo cenna właściwość ze względu na znaczne skrócenie czasu niezbędnego do pełnego przetestowania „wymyślonej" przez nas koncepcji.

Wszystkie przedstawione dalej programy są przeznaczone do testowania projektów opublikowanych w EP 12/93 - sześciu funktorów logicznych, dekodera wyświetlacza 7-segmentowego oraz licznika dekadowego.

Procedura testująca realizowana przez CUPL'a opiera się na pliku *.SI, w którym projektant musi zdefiniować:

- wejściowe wektory testujące - muszą się tam znaleźć wszystkie kombinacje sygnałów wejściowych, których wpływ na pracę układu chcemy przeanalizować;
- wyjściowe wektory testowe, czyli stany logiczne wyjść w odpowiedzi na odpowiednie pobudzenie (wektorem wejściowym), które chcemy przeanalizować. Ten fragment nie jest niezbędny - omówimy to za chwilę.

Oprócz pliku wejściowego z zadanymi wektorami testowymi do poprawnej symulacji niezbędny jest dodatkowy plik *.ABS. Jest to plik wytwarzany przez kompilator CUPL w czasie kompilacji projektu (*.PLD). Zawarta jest w tym pliku binarna wersja realizowanego projektu.

Wynikiem pracy programu CSIM jest plik wyjściowy *.SO, w którym zawarte są wszystkie uwagi i sygnalizacje ewentualnych błędów wykrytych w czasie symulacji, plik *.JED z wektorami testowymi (przykład na **list. 1**), który jest wykorzystywany do testowania zaprogramowanego układu (fizycznej „kostki") oraz plik *.PRT zawierający najbardziej znaczące przebiegi czasowe (odniesione do numeru wektora testującego lub taktu zegara - w układach sekwencyjnych).

Na **rys. 1** przedstawiony jest obieg informacji w symulatorze CSIM, przy czym pominięto kompilator CUPL (założono, że plik *.ABS jest dostępny).

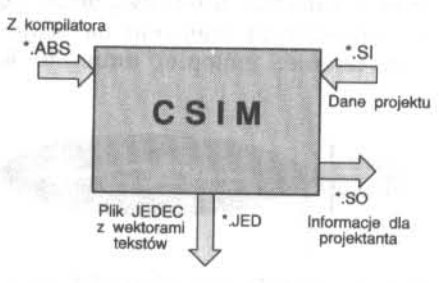
Jako ciekawostkę można podać fakt, że w starszych wersjach CUPL'a wywoływanie symulatora (podobnie jak i kompilatora) odbywało się z poziomu DOS'a. Każdy z programów należało wywołać z odpowiednimi parametrami, w zależności od oczekiwanych efektów pracy programu. W wersji CUPL'a oferowanej przez AVT wszystkie selekcje przeprowadza się z poziomu MAIN MENU programu, ale istnieje także możliwość wywoływania poszczególnych fragmentów kompilatora niezależnie (tak jak w starszych wersjach). Na końcu artykułu podane zostaną niezbędne parametry wraz z ich krótkim opisem, z którymi można wywołać program CSIM.

```
CUPL            4.0a  Serial# MD-40A-8209
Device          g16v8s  Library DLIB-h-26-9
Created         Tue Dec 14 16:55:22 1993
Name            Bramki
Partno          2
Revision        1.0
Date            11/14/93
Designer        P.Z.
Company         BTC
Assembly        xxxxxx
Location        US1
*QP20
*QF2194
*QV4
*G0
*F0
*L00000  01011111111111111111111111111111
*L00256  11110111011111111111111111111111
*L00512  11111111111101111111111111111111
*L00544  11111111111111101111111111111111
*L00768  11111111111111111101111111111111
*L00800  11111111111111111111101111111111
*L01024  11111111111111111111111111110110
*L01056  11111111111111111111111111111001
*L01280  11111111111111111111011011111111
*L01312  11111111111111111111011101111111
*L02048  10101000010011000000000000000000
*L02112  00000000000000011111111111111111
*L02144  11111111111111111111111111111111
*L02176  11111111111111110
*C2EE4
*P 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
*V0001  000000000N000HLHLHLN
*V0002  01010101ON101LHLHHLN
*V0003  101010101N010LHLHHLN
*V0004  111111111N111HLLHLHN
*DB0A
```

List. 1. Plik *.JED z wektorami testującymi

Struktura pliku testowego oraz zasady opisu projektu

Na **list. 2** przedstawiono podstawową strukturę pliku *.SI. Składa się on z nagłówka (takiego samego jak w pliku *.PLD), pola deklaracji kolejności wektorów testowych (opis rozpoczyna się od słowa ORDER) oraz pola definiowania wartości tych wektorów (początek po słowie VECTORS). W obydwu tych polach określa się wartości sygnałów wej-



Rys. 1. Obieg informacji w symulatorze CSIM

```
Name     Przyklad;
Partno   4;
Date     11/14/93;
Revision 1.99;
Designer P.Z.;
Company  BTC;
Location US167;
Device   G16V8;

/* Powyzej znajduje sie standardowy naglowek     */


ORDER:  X,Y,Z,A3..0,Wy1..6;
/* Powyzej znajduje sie definicja                */
/* kolejnosci wektorow testowych                 */


VECTORS:
/* Powyzej rozpoczyna sie pole deklaracji        */
/* wektorow testowych. Uporzadkowanie wg         */
/* kolejnosci ustalonej w deklaracji ORDER       */

/* Ponizej znajduje sie polecenie wyswietlenia   */
/* informacji pomocniczej                        */
$MSG"Tresc komunikatu";

/* Od tego miejsca zaczynaja sie wektory testowe */
0000000LLL
0000001HLH
1001101HHH
XX11010***
```

List. 2. Struktura pliku *.SI

```
Name     List_3;
Partno   1;
Date     11/14/93;
Revision 1.0;
Designer P.Z.;
Company  BTC;
Location US1;
Device   G16V8;

/* Deklaracja kodu liczbowego w ktorym zapisane */
/* beda dane */
BASE: hex;

/* Deklaracja kolejnosci wektorow testowych      */
ORDER:   AO..5,%1,Y1,
         A,B,%1,Y6;


/* Definicje kolejnych wektorow testowych        */

VECTORS:
000X00L11H
1001XXL10H
111X10H01H
100000H00L
```

List. 3. Sposób zastosowania deklaracji listowych i deklaracji BASE.

```
Name     Bramki;
Partno   2;
Date     11/14/93;
Revision 1.0;
Designer P.Z.;
Company  BTC;
Location US1;
Device   G16V8;

/* ***************************************** */
/*       Plik symulacyjny do projektu        */
/*     szeciu bramek logicznych (EP12/93)    */
/*   WERSJA PLIKU BEZ JAWNEGO ZADKLAROWANIA  */
/*            STANU WYJSC                    */
/* ***************************************** */

/* Deklaracja kolejnosci wektorow testowych      */

ORDER:   A,B,%1,Y1,
         %3,C,D,%1,Y2,
         %3,E,F,%1,Y3,
         %3,G,H,%1,Y4,
         %3,I,J,%1,Y5,
         %3,K,L,%1,Y6;


/* Definicje kolejnych wektorow testowych        */

VECTORS:
$MSG"  AND * NAND *  OR  *  NOR *  ExR * ExNR";
$MSG"  AB Y1  CD Y2  EF Y3  GH Y4  IJ Y5  KL Y6";

00*00*00*00*00*00*  /* Na wszystkie wej.podano 00 */
01*01*01*01*01*01*  /* Na wszystkie wej.podano 01 */
10*10*10*10*10*10*  /* Na wszystkie wej.podano 10 */
11*11*11*11*11*11*  /* Na wszystkie wej.podano 11 */
```

List. 4. Pierwsza wersja pliku testowego do weryfikacji projektu BRAMKI.PLD

ściowych i wyjściowych. Oznaczenia stosowane do deklarowania kolejności sygnałów wejściowych i wyjściowych muszą się pokrywać z oznaczeniami stosowanymi w opisie projektu (pliku *.PLD), np. jeżeli w pliku opisującym bramki logiczne BRAMKI.PLD (EP12/93) jako wejścia bramki AND zadeklarowano zmienne A oraz B, a wyjście oznaczono Y1, to program symulacyjny korzystający z pliku BRAMKI.SI (**list. 4**) zinterpretuje zmienne A, B, Y1 w taki sam sposób. Blok deklaracji kolejności składowych wektora testowego musi być zakończony średnikiem.

Wszystkie sygnały (wejściowe i wyjściowe) można podawać w formie dyskretnej (jawne wymienienie nazwy wszystkich wektorów) lub formie deklaracji pól bitowych z indeksami, przykład znajduje się na **list. 3**.

Deklarowane zmienne mogą być podawane w różnych systemach liczbowych (dwójkowy, ósemkowy, dziesiętny, szesnastkowy), przy czym, dzięki zastosowaniu słowa BASE, nie jest konieczne każdorazowe podawanie systemu liczenia (przykład na list. 3).

Sygnałom wejściowym, interpretowanym jako pobudzenia, można przyporządkować następujące wartości:

1 - wysoki poziom logiczny (fizycznie oznacza to podanie wysokiego (TTL) poziomu napięcia na to wejście);

0 - niski poziom logiczny (podanie niskiego (TTL) poziomu napięcia na testowane wejście);

X - poziom nieistotny (wpływ tego wejścia jest pominięty w analizie pracy układu);

K - sygnał zegarowy, synchronizacja zboczem opadającym;

C - sygnał zegarowy, synchronizacja zboczem narastającym;

Stany wyjściowe badanego układu można zdefiniować na dwa sposoby:

Przyporządkowanie wektorowi wejściowemu odpowiadającego mu wektora na wyjściu. Do opisu stosowane są wtedy następujące oznaczenia:

H - wysoki poziom logiczny (wysoki poziom napięcia w standardzie TTL);

L - niski poziom logiczny (niski poziom napięcia w standardzie TTL);

Z - stan wysokiej impedancji (brak napięcia na wyjściu).

W tym wypadku symulator sam wskazuje błędne stany (odpowiedzi układu różne od zadanych), co znakomicie upraszcza testowanie, ale jest okupione większym nakładem pracy na opracowanie pliku wejściowego (z praktyki autora wynika, że bardzo trudno jest uniknąć tu błędów).

Można zrezygnować z definiowania oczekiwanej odpowiedzi układu na pobudzenie wejściowe, wstawiając w miejscu analizowanego sygnału symbol *. W tak zadeklarowane pola symulator sam wstawi sygnał będący reakcją układu na stany logiczne podane na wejścia. W pewnych sytuacjach takie postępowanie jest wygodniejsze, należy jednakże pamiętać o konieczności „ręcznej" weryfikacji wyników symulacji. Można to zrobić w oparciu o plik wyjściowy symulatora *.SO lub wykresy czasowe z pliku *.PRT.

Dla uwidocznienia różnic w konstrukcji tych plików (z definiowaniem oczekiwanej odpowiedzi i bez niej) na **list. 4** i **list. 5** (pliki symulacyjne do testowania funktorów logicznych) przedstawiono pliki testujące ten sam projekt na dwa różne sposoby. Jak widać, różnica w zapisie jest niewielka, nieco trudniej jest to zrobić dla bardziej złożonych projektów.

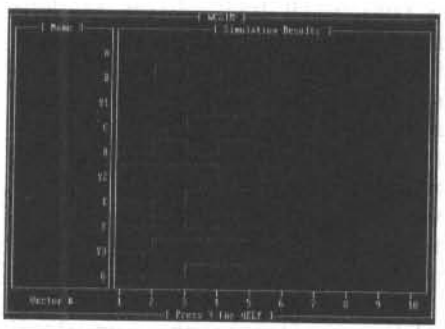
Listing 6 przedstawia zawartość pliku wyjściowego symulacji BRAMKI.SO. Niezależnie od sposobu definiowania stanów wyjściowych wyniki symulacji są takie

```
Name     Bramki;
Partno   2;
Date     11/14/93;
Revision 1.0;
Designer P.Z.;
Company  BTC;
Location US1;
Device   G16V8;

/* ***************************************** */
/*       Plik symulacyjny do projektu szeciu     */
/*         bramek logicznych (EP12/93)           */
/*  WERSJA PLIKU Z JAWNA DEKLARACJA STANU WYJSC  */
/* ***************************************** */

/* Deklaracja kolejnosci wektorow testowych      */

ORDER:   A,B,%1,Y1,
         %3,C,D,%1,Y2,
         %3,E,F,%1,Y3,
         %3,G,H,%1,Y4,
         %3,I,J,%1,Y5,
         %3,K,L,%1,Y6;


/* Definicje kolejnych wektorow testowych        */

VECTORS:
$MSG"      AND * NAND *  OR  *  NOR *  ExR * ExNR";
$MSG"    AB Y1  CD Y2  EF Y3  GH Y4  IJ Y5  KL Y6";

00L00H00L00H00L00H  /* Na wszystkie wej.podano 00 */
01L01H01H01L01H01L  /* Na wszystkie wej.podano 01 */
10L10H10H10L10H10L  /* Na wszystkie wej.podano 10 */
11H11L11H11L11L11H  /* Na wszystkie wej.podano 11 */
```

List. 5. Druga wersja pliku testowego do weryfikacji projektu BRAMKI.PLD

```
CSIM: CUPL Simulation Program
Version 4.0a Serial# MD-40A-8209
Copyright (C)  1983,1990 Logical Devices, Inc.
CREATED Tue Dec 14 10:38:58 1993

LISTING FOR SIMULATION FILE: C:\CUPL40\BRAMKI.si

   1: Name      Bramki;
   2: Partno    2;
   3: Date      11/14/93;
   4: Revision  1.0;
   5: Designer  P.Z.;
   6: Company   BTC;
   7: Location  US1;
   8: Device    G16V8;
   9:
  10:
  11:
  12:
  13:
  14:
[0026sa] Please note: missing header item(s)
  15: ORDER:   A,B,%1,Y1,
  16:        %3,C,D,%1,Y2,
  17:        %3,E,F,%1,Y3,
  18:        %3,G,H,%1,Y4,
  19:        %3,I,J,%1,Y5,
  20:        %3,K,L,%1,Y6;
  21:
=============================================
            Simulation Results
=============================================
     AND * NAND *  OR  *  NOR *  ExR * ExNR
     AB Y1  CD Y2  EF Y3  GH Y4  IJ Y5  KL Y6
0001: 00 L  00 H  00 L  00 H  00 L  00 H
0002: 01 L  01 H  01 H  01 L  01 H  01 L
0003: 10 L  10 H  10 H  10 L  10 H  10 L
0004: 11 H  11 L  11 H  11 L  11 L  11 H
```

List. 6. Plik wyjściowy symulatora CSIM po wykonaniu programu z list. 4.



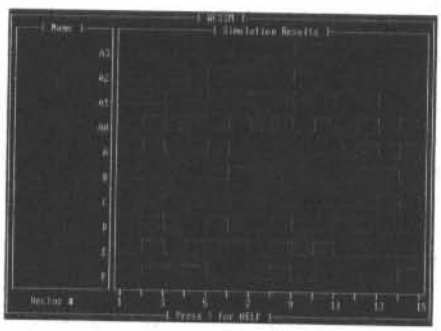
Rys. 2. Widok ekranu symulatora z przebiegami dla pliku testującego bramki logiczne

same. Na **rys. 2** znajduje się widok ekranu komputera podczas symulowania pliku z list. 4 - widać przebiegi czasowe (na osi czasu znajduje się numer wektora testującego). Przebiegi te odpowiadają stanom wyprowadzonym w pliku wyjściowym. Symulację oparto na zasadzie kolejnego podawania, jednocześnie dla wszystkich bramek, wszelkich możliwych kombinacji stanów sygnałów wejściowych (jest ich 4).

Na **list. 7** znajduje się plik symulacyjny wykorzystywany do testowania dekodera wyświetlacza BCD->7-segm. HEX. Podobnie jak w poprzednim przypadku na wejścia danych A3...0 podawane są wszystkie możliwe kombinacje sygnałów wejściowych (jest ich 16) i następuje porównywanie odpowiedzi obliczonej z zadeklarowaną. **Listing 8**

```
PARTNO US2;
NAME DeC_HEX;
REV 1.4;
DATE 12/10/93;
DESIGNER Piotr Zbysinski;
ASSEMBLY N/A;
COMPANY BTC;
LOCATION N/N;
DEVICE G16V8;

/* ***************************************** */
/*    Program symulacyjny dla dekodera kodu  */
/*    dwojkowego na kod wyswietlacza 7-segm. */
/* ***************************************** */

/* Deklaracja kolejnosci wektorow testowych  */

ORDER:A3..A0,%2,A,B,C,D,E,F,G;


/* Definicje wektorow testowych              */

VECTORS:
$MSG"       AAAA  ABCDEFG";
$MSG"       3210";
0000LLLLLLH
0001HLLHHHH
0010LLHLLHL
0011LLLLHHL
0100HLLHHLL
0101LHLLHLL
0110LHLLLLL
0111LLLHHHH
1000LLLLLLL
1001LLLLHLL
1010LLLLHLL
1011HHLLLLL
1100LHHLLLL
1101HLLLLHL
1110LHHLLLL
1111LHHHLLL
```

List. 7. Program testowy dekodera BCD->7-segm. Hex



Rys. 3. Zależności czasowe sygnałów dla dekodera BCD->7-segm. HEX

prezentuje wynik symulacji za pomocą pliku DeC_HEX.SI (zależności czasowe wejście-wyjście znajdują się na **rys. 3**, na osi czasu znajduje się numer wektora testowego).

Listing 9 przedstawia plik symulacyjny dla licznika dekadowego. Test podzielono na trzy zasadnicze etapy:

- wstępne zerowanie (pierwsze dwa takty zegara - pierwsze dwie linie po słowie VECTORS). Utrzymywanie kasowania przez dwa takty wprowadzono w celu pewnego działania testu - do poprawnego skasowania licznika wystarczy tylko jeden takt zegara (w projekcie LICZNIK.PLD wejście kasujące zadeklarowano jako synchroniczne!);
- liczenie w górę (od 0 do 9). Oczekiwane stany wyjściowe podane zostały w sposób jawny (kolejno

```
CSIM: CUPL Simulation Program
Version 4.0a Serial# MD-40A-8209
Copyright (C)  1983,1990 Logical Devices, Inc.
CREATED Tue Dec 14 10:51:32 1993

LISTING FOR SIMULATION FILE: C:\CUPL40\HEX.si

   1: PARTNO US2;
   2: NAME DeC_HEX;
   3: REV 1.4;
   4: DATE 12/10/93;
   5: DESIGNER Piotr Zbysinski;
   6: ASSEMBLY N/A;
   7: COMPANY BTC;
   8: LOCATION N/N;
   9: DEVICE G16V8;
  10:
  11: ORDER:A3..A0,%2,A,B,C,D,E,F,G;
  12:
=============================================
            Simulation Results
=============================================
      AAAA  ABCDEFG
      3210
0001: 0000  LLLLLLH
0002: 0001  HLLHHHH
0003: 0010  LLHLLHL
0004: 0011  LLLLHHL
0005: 0100  HLLHHLL
0006: 0101  LHLLHLL
0007: 0110  LHLLLLL
0008: 0111  LLLHHHH
0009: 1000  LLLLLLL
0010: 1001  LLLLHLL
0011: 1010  LLLHLLL
0012: 1011  HHLLLLL
0013: 1100  LHHLLLL
0014: 1101  HLLLLHL
0015: 1110  LHHLLLL
0016: 1111  LHHHLLL
```

List. 8. Plik wyjściowy symulatora po wykonaniu programu z list. 7.

wpisano oczekiwane w danym momencie stany wyjściowe);

- liczenie w dół (poprzedzone jednym taktem kasującym - bez wpływu na stany wyjść). Tutaj zastosowano inną formę zapisu - szybszą, lecz nieco trudniejszą w analizie. Procedura zliczania w dół rozpoczyna się od słowa kluczowego $REPEAT 10. Rozkaz ten zostaje zinterpretowany przez symulator jako 10-krotne powtórze-

```
Name      Licznik;
Partno    US3;
Date      28/10/93;
Revision  1.9;
Designer  P.Z.;
Company   BTC;
Device    g16v8;

/* ***************************************** */
/* Program symulacyjny dla licznika dziesietnego */
/* ***************************************** */

/* Deklaracja kolejnosci elementow           */
/* wektora testowego                         */

ORDER:  CLK,RES,DIR,
        Q3..0,
        CARRYO;

/* Definicja wektorow testowych                    */
/* Ponizszy program wykonuje                       */
/* nastepujaca sekwencje testow:                   */
/* - kasowanie licznika (dwa takty zegara);        */
/* - cykl zliczania w gore (od 0 do 9);            */
/* - kasowanie licznika (jeden takt zegarowy);     */
/* - cykl zliczania w dol (od 9 do 0);             */

VECTORS:
C1XLLLLL
C1XLLLLL
C0OLLLHL
C0OLLHLL
C0OLLHHL
C0OLHLLL
C0OLHLHL
C0OLHHLL
C0OLHHHL
C0OHLLLL
C0OHLLHH
C1XLLLLL
$REPEAT 10;
C01*****
```

List. 9. Plik testowy dla licznika dekadowego

```
CSIM: CUPL Simulation Program
Version 4.0a Serial# MD-40A-8209
Copyright (C)  1983,1990 Logical Devices, Inc.
CREATED Tue Dec 14 11:05:06 1993

LISTING FOR SIMULATION FILE: C:\CUPL40\LICZN.si

   1: Name     Licznik;
   2: Partno   US3;
   3: Date     28/10/93;
   4: Revision 1.9;
   5: Designer P.Z.;
   6: Company  BTC;
   7: Device   gl6v8;
   8:
[0026sa] Please note: missing header item(s)
   9: ORDER: CLK,RES,DIR,
  10:       Q3..0,
  11:       CARRYO;
  12:
=================================================
                Simulation Results
=================================================
0001: C1XLLLLL
0002: C1XLLLLL
0003: C00LLLHL
0004: C00LLHLL
0005: C00LLHHL
0006: C00LHLLL
0007: C00LHLHL
0008: C00LHHLL
0009: C00LHHHL
0010: C00HLLLL
0011: C00HLLHH
0012: C1XLLLLL
0013: C01HLLHL
0014: C01HLLLL
0015: C01LHHHL
0016: C01LHHLL
0017: C01LHLHL
0018: C01LHLLL
0019: C01LLHHL
0020: C01LLHLL
0021: C01LLLHL
0022: C01LLLLH
```

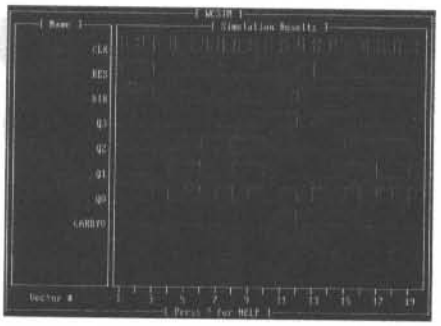
List. 10. Plik wyjściowy symulatora, po wykonaniu programu z list. 9

nie sekwencji następującej bezpośrednio po słowie kluczowym. Inaczej mówiąc, do wirtualnego modelu układu dostarczone będzie 10 taktów zegara (narastające zbocza C), przy stałym „0" na wejściu RES i stałej „1" na wejściu DIR. Po każdym takcie nastąpi obliczenie odpowiedzi układu (czyli stanów wyjściowych) i zapamiętanie jej. Jest to metoda zdecydowanie szybsza, ale wymagająca znacznej uwagi przy weryfikacji otrzymanych wyników. Na **rys. 4** znajduje się widok ekranu komputera z wynikami pracy symulatora CSIM. **Listing 10** zawiera tekstowy zapis wyników symulacji.

Na pewno uważniejsi Czytelnicy zauważyli na wszystkich listingach plików wejściowych kilka szczegółów, o których wcześniej nie wspomniano. Są to:

- znaki %1, %3 (ogólnie N) - określają, ile spacji (1, 3, ogólnie N) symulator powinien wstawić pomiędzy zadeklarowane pola w pliku *.SO. Są to znaki porządkowe i nie mają żadnego wpływu na proces symulacji;

- polecenie $MSG"Tekst", które, umieszczone po słowie kluczowym VECTORS, umożliwia wyświetlenie nagłówka (lub dowolnego innego tekstu) nad wynikami symulacji. Efekt tego polecenia można zaobserwować w plikach wyjściowych



Rys. 4. Zależności czasowe dla układu licznika dekadowego

(wszystkie nagłówki nad stanami wyjściowymi powstały dzięki zastosowaniu dyrektywy $MSG). Linię zawierającą tego typu polecenie należy zakończyć średnikiem. Polecenie to ma znaczenie porządkowe, nie wpływa w żaden sposób na wyniki symulacji.

Symulator dopuszcza także stosowanie komentarzy rozpoczynających się od pary znaków /*, a kończących się parą */ (przykład w list.1), identycznie jak program kompilujący.

Wymienione do tej pory słowa sterujące (dyrektywy) symulatora są najczęściej stosowane. Oprócz nich symulator dopuszcza stosowanie następujących dyrektyw:

$SIMON i $SIMOFF - służą do sterowania wyliczaniem wartości wektorów testujących. Mogą służyć do miejscowego wyłączenia symulacji w pliku *.SI;

$EXIT - rozkaz przerwania symulacji od miejsca, w którym jest on wstawiony.

Parametry programu CSIM i ich znaczenie

Wywołanie symulatora (z poziomu DOS'a):

CSIM parametr [biblioteka] [nazwa układu] nazwa_pliku

gdzie parametr może przyjąć wartość:

L - zapisanie pliku z wynikami pracy symulatora (Listing File);

J - dodanie do pliku *.JED (z danymi dla programatora) zbioru wektorów testujących;

D - wyświetlenie przebiegów czasowych na ekranie komputera;

W - wyświetlenie przebiegów czasowych na ekranie i zapisanie ich do pliku;

V - bieżące wyświetlanie wyników symulacji;

U - wykorzystanie specyficznych bibliotek (odmiennych niż w czasie kompilacji) przez symulator.

[biblioteka] - nazwa biblioteki układów PLD (standardowo jest to CUPL.DL, w pliku AUTOEXEC.BAT powinien znajdować się zapis

SET LIBCUPL=napęd:{ścieżka dostępu}\CUPL.DL

[nazwa układu] - typ układu dla którego kompilowany jest projekt (np. gl6v8, pl12p6, itp.).

Przy pracy z poziomu shell'a program CSIM otrzymuje wszystkie parametry automatycznie, po wybraniu przez projektanta z menu interesujących go opcji.

Piotr Zbysiński, AVT

Literatura:

1. CUPL 4.0 Quick Reference. Logical Devices, 1992.
2. CUPL - User Manual. Logical Devices, 1990.
3. Programowalne moduły logiczne w syntezie układów cyfrowych. W. Majewski, T. Łuba, B. Zbierzchowski. WKiŁ, 1992.