Układy programowalne, część 8

Zbliżamy się do końca cyklu, więc po sporej dawce informacji o języku CUPL, przechodzimy do przedstawienia narzędzi, w których ten właśnie język HDL został zaimplementowany. W tej części przedstawiamy możliwości bezpłatnych programów: edytora schematów RimuSCH i kompilatora WinCUPL.

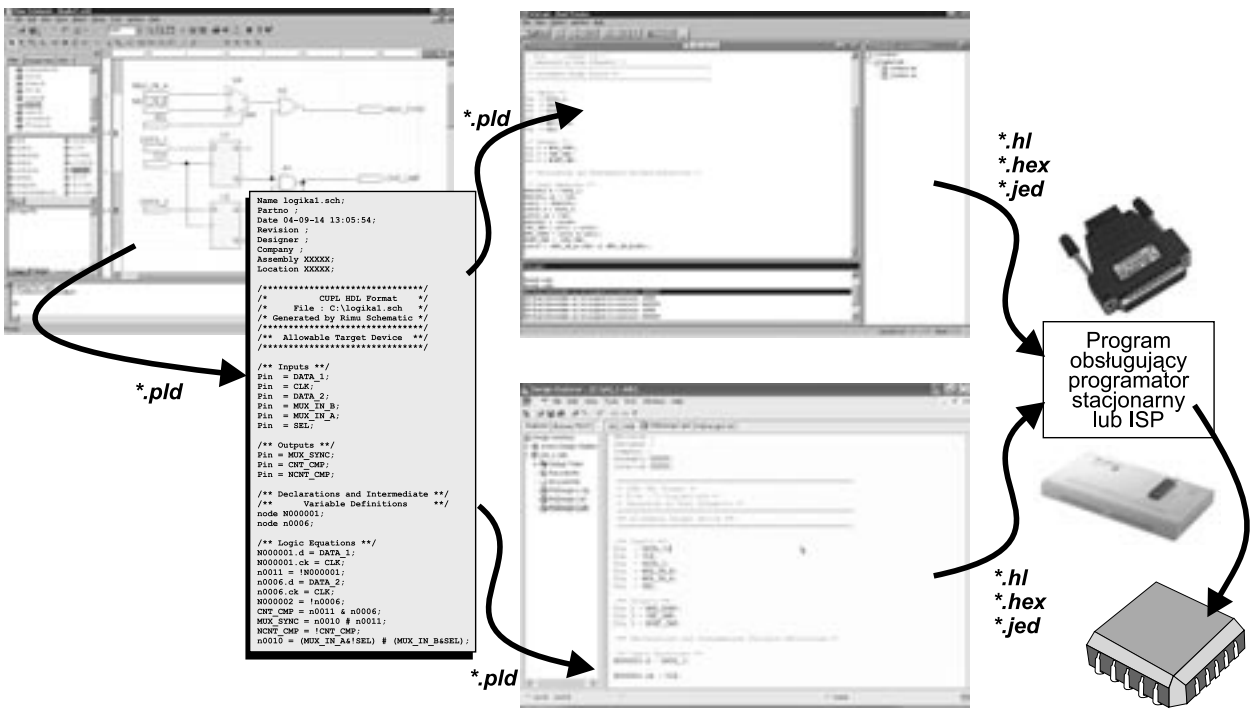
„Plotki o mojej śmierci są nieco przesadzone" powiedział niegdyś Mark Twain. Powiedzenie to doskonale pasuje do obecnej sytuacji CUPL-a: z mody nieco wyszedł, ale jest ciągle stosowany m.in. w pakietach projektowych Protel (także w najnowszej wersjach DXP i 2004). Warto zwrócić uwagę na fakt, że kompilatory wbudowane w pakiety firmy Altium (Protel 99SE/DXP/2004) obsługują blisko 6000 typów układów SPLD/CPLD oraz ponad 8500 typów układów FPGA pochodzących od 20 producentów. CUPL jest także sztandarowym językiem HDL firmy Atmel, która udostępnia na swojej stronie internetowej bezpłatną wersję WinCUPL-a. Kompilator CUPL-a zastosowano także w pakiecie ProChip Designer firmy Atmel, który służy m.in. do realizacji projektów na układy FPSLIC (rdzeń AVR i FPGA w jednej obudowie).

Do grona producentów narzędzi ułatwiających korzystanie z CUPL-a dołączyła także nowozelandzka firma Hutson (*http://www.hutson.co.nz*), w ofercie której znajduje się program Rimu Schematic. Za jego pomocą można konwertować schematy elektryczne do postaci opisu tekstowego w języku CUPL. Program ten nie ma wbudowanego kompilatora, nie może on więc pracować całkowicie samodzielnie – służy wyłącznie jako wygodny konwerter schematów do opisu HDL. Prezentację narzędzi rozpoczniemy od tego właśnie programu, którego możliwości w bezpłatnej wersji są w zupełności wystarczające do zrealizowania projektów na układach GAL i większych.

RimuSCH:
dla tych co nie lubią pisać

Pomimo tego, że CUPL należy do zdecydowanie najprostszych języków HDL, wielu jego potencjalnych użytkowników obawia się podejmowania samodzielnych prób z przygotowywaniem za jego pomocą własnych opisów projektowanych układów. Uważają (i słusznie), że łatwiej byłoby narysować schemat logiczny, który jakieś narzędzie przekonwertuje do postaci „zrozumiałej" dla programowanego układu PLD. Z pomocą przychodzi nam RimuSCH – dostępny bezpłatnie (publikujemy go na CD-EP10/2004B) edytor schematów wyposażony m.in. w interfejs eksportu listy połączeń w formacie CUPL-a. Program wymaga instalacji, która przebiega w sposób typowy dla systemu Windows (działa także z Windows XP).

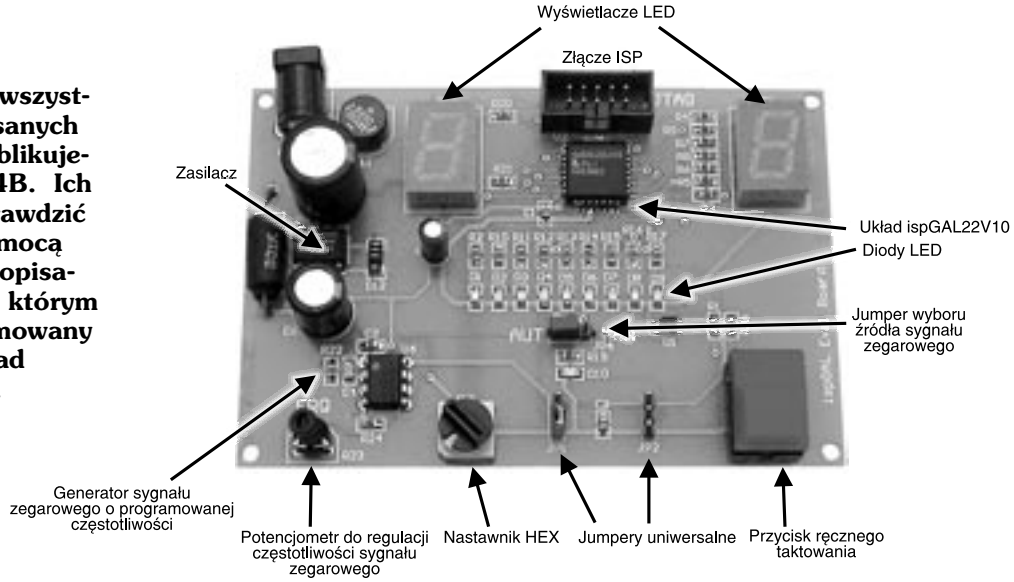
Na **rys. 39** pokazano ulokowanie RimuSCH w typowym cyklu projektowym. Rolę syntezera logicznego może spełniać dowolny kompilator CUPL-a, także jego wersja DOS-owa. Na rysunku przedstawiono dwa zalecane programy:



Rys. 39

Można także w praktyce

Programy źródłowe wszystkich projektów opisanych w ramach kursu publikujemy na CD-EP9/2004B. Ich działanie można sprawdzić w praktyce za pomocą zestawu AVT-599 (opisany w EP3/2004), w którym zastosowano programowany w systemie układ ispGAL22V10.



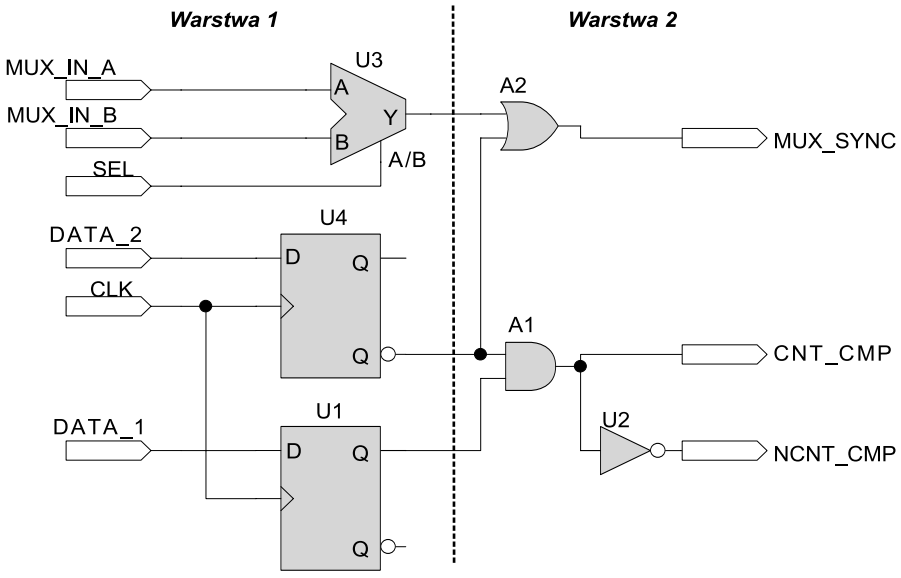
– WinCUPL, którego najpoważniejszą (w zasadzie jedyną) wadą są mocno ograniczone biblioteki układów docelowych (do układów zgodnych z oferowanymi przez Atmela), co jest o tyle oczywiste, że ta wersja programu jest własnością Atmela. Niebagatelną zaletą tego programu jest możliwość nieodpłatnego (przy tym legalnego) korzystania z niego.
– Protel 99SE lub nowszy, którego wadą jest ograniczony czas legalnego korzystania (ze względu na ograniczenia wersji ewaluacyjnej), za to biblioteki obsługiwanych układów są niezwykle bogate.

Tak więc, kompilator można dobrać do indywidualnych potrzeb i wymagań. W przypadku projektów prezentowanych w tym cyklu artykułów, do ich przetestowania w zestawie AVT-599 w zupełności wystarczy WinCUPL.

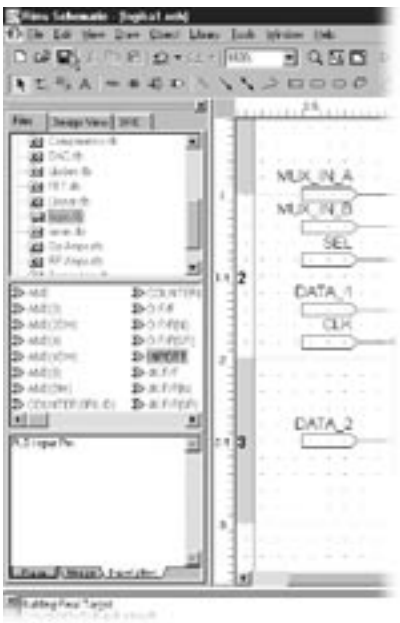
Projekt w dwóch plikach

Kompletny projekt CUPL składa się z dwóch plików o rozszerzeniach: **.pld* (źródłowy) oraz **.si* (symulacyjny - jeżeli symulator jest wykorzystywany podczas projektowania). Przeniesienie projektu pomiędzy programami wykorzystywanymi podczas projektowania oznacza w praktyce przeniesienie tych dwóch plików.

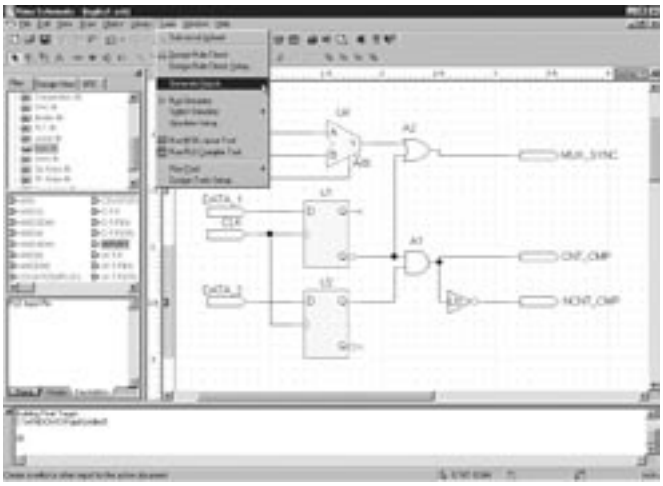
Na **rys. 40** pokazano schemat logiczny przykładowego projektu, który przygotowano za pomocą edytora RimuSCH. Podczas rysowania schematów logicznych przeznaczonych do konwersji do postaci HDL, należy korzystać wyłącznie z biblioteki *logic.rlb*, która jest dostarczana w standardowej instalacji edytora (**rys. 41**). W jej ramach są dostępne wszystkie standardowe funktory logiczne (bram-



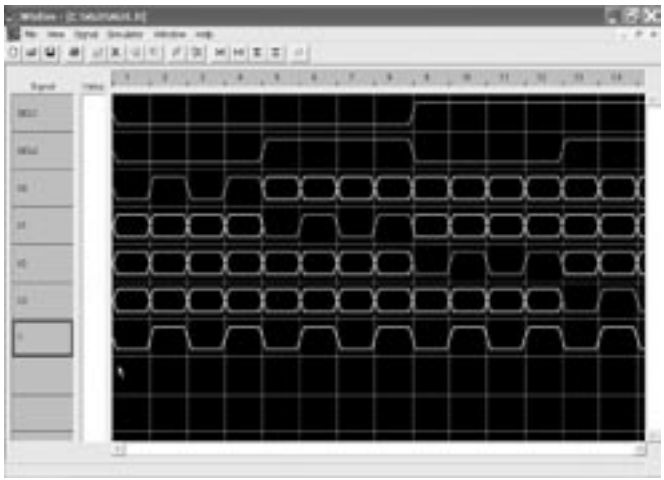
Rys. 40



Rys. 41



Rys. 42



Rys. 44

ki AND, NAND, OR, NOR, ExOR i ExNOR, także w wersjach o dużej liczbie wejść, ich symbole są dostępne w dwóch notacjach), przerzutniki JK, T, D i SR, a także multipleksery. Twórca RimuSCH zawarł ponadto w bibliotece *logic.rlb* dwa złożone elementy biblioteczne – 8-bitowe liczniki góra-dół z wyjściami trójstanowymi, które – ze względu na przyjęty sposób opisu – nie we wszystkich układach PLD dają się wygodnie zaimplementować.

Po narysowaniu schematu można go poddać weryfikacji elektrycznej za pomocą standardowego narzędzia ERC wbudowanego w RimuSCH (pozwala wychwycić „grube" pomyłki w narysowanym schemacie), a następnie wyeksportować do formatu pliku źródłowego **.pld* z opisem w języku CUPL. Wymaga to wybrania w menu opcji *Tools> Generale Report...* (**rys. 42**), następnie w wyświetlonym oknie (**rys. 43**) wybieramy format pliku wyjściowego (na liście *Report Type*). W ten sposób uzyskujemy plik źródłowy z opisem projektu przedstawionego na schemacie (rys. 40) w języku CUPL. Opis HDL przedstawiono na **list. 19**. Niestety, pokazany przykład ilustruje drobną słabość wbudowanego w RimuSCH generatora plików wyjściowych: w przypadku projektów wielopoziomowych (tzn. takich, w których występują zagrzebane węzły logiczne wykorzystywane do implementacji funkcji logicznych – rys. 40) edytor samodzielnie tworzy nazwy węzłów (jak *N000001* lub *n0006* na list. 19), ale nie deklaruje ich w pliku źródłowym. Powoduje to, że wygenerowany plik nie daje się skompilować. Konieczne jest uzupełnienie go o deklaracje węzłów zagrzebanych:

```
node N000001;
node n0006;
```

lub węzłów przypisanych do wyprowadzeń docelowego układu:

```
pinnode = N000001;
pinnode = n0006;
```

Rys. 43

Kompilacja opisu – możliwość pierwsza

Uzyskany opis poddamy kompilacji za pomocą programu WinCUPL. Ponieważ RimuSCH generuje plik w formacie **.pld*, sformatowany zgodnie ze specyfikacją CUPL-a, można wykorzystać go jak standardowy plik projektowy. W przypadku takiej konieczności, opis uzyskany z RimuSCH należy zmodyfikować o wcześniej wspomniane deklaracje, co umożliwi jego kompilację.

W zależności od potrzeb, WinCUPL umożliwia kompilację docelową, tzn. z uwzględnieniem ograniczeń architektury fizycznych układów lub – co jest przydatne podczas weryfikacji poprawności opisu HDL – kompilację na układ wirtualny (*virtual device*), który nie ma żadnych ograniczeń wynikających z budowy jego mikrokomórek, buforów I/O itp. Podczas pierwszych samodzielnych prób zdecydowanie łatwiejsze będzie

Nie do końca doskonały

RimuSCH generuje niepoprawne opisy dla projektów składających się z więcej niż jednej warstwy. Konieczne jest uzupełnienie wygenerowanego opisu o deklaracje węzłów zagrzebanych (*node*) oraz wyprowadzonych na zewnątrz układu (*pinnode*).

List. 19. Plik źródłowy z opisem projektu pokazanego na rysunku 40, uzyskany za pomocą edytora RimuSCH

```
Name logika1.sch;
Partno ;
Date 04-09-14 13:05:54;
Revision ;
Designer ;
Company ;
Assembly XXXXX;
Location XXXXX;

/*******************************/
/*       CUPL HDL Format       */
/*    File : C:logika1.sch     */
/* Generated by Rimu Schematic */
/*******************************/
/**  Allowable Target Device  **/
/*******************************/

/** Inputs **/
Pin  = DATA_1;
Pin  = CLK;
Pin  = DATA_2;
Pin  = MUX_IN_B;
Pin  = MUX_IN_A;
Pin  = SEL;

/** Outputs **/
Pin = MUX_SYNC;
Pin = CNT_CMP;
Pin = NCNT_CMP;

/** Declarations and Intermediate Variable
Definitions */

/** Logic Equations **/
N000001.d = DATA_1;

N000001.ck = CLK;

n0011 = !N000001;
n0006.d = DATA_2;

n0006.ck = CLK;

N000002 = !n0006;
CNT_CMP = n0011 & n0006;
MUX_SYNC = n0010 # n0011;
NCNT_CMP = !CNT_CMP;
n0010 = (MUX_IN_A&!SEL) # (MUX_IN_B&SEL);
```

List. 20. Plik zawierający opis symulacji multipleksera z list. 10 (EP8/2004)

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

ORDER: SEL1, SEL0, X0, X1, X2, X3, Y;
VECTORS:
000XXX*
001XXX*
000XXX*
001XXX*
01X0XX*
01X1XX*
01X0XX*
01X1XX*
10XX0X*
10XX1X*
10XX0X*
10XX1X*
11XXX0*
11XXX1*
11XXX0*
11XXX1*
```

List. 21. Pliki z wynikami symulacji z formatowaniem i bez niego

```
Bez formatowania
================
     SS
     EE
     LLXXXX
     100123Y
================
0001: 000XXXL
0002: 001XXXH
0003: 000XXXL
0004: 001XXXH
0005: 01X0XXL
0006: 01X1XXH
0007: 01X0XXL
0008: 01X1XXH
0009: 10XX0XL
0010: 10XX1XH
0011: 10XX0XL
0012: 10XX1XH
0013: 11XXX0L
0014: 11XXX1H
0015: 11XXX0L
0016: 11XXX1H

Z formatowaniem
=====================
     SS
     EE
     LL    XXXX
     10    0123  Y
=====================
0001: 00    0XXX  L
0002: 00    1XXX  H
0003: 00    0XXX  L
0004: 00    1XXX  H
0005: 01    X0XX  L
0006: 01    X1XX  H
0007: 01    X0XX  L
0008: 01    X1XX  H
0009: 10    XX0X  L
0010: 10    XX1X  H
0011: 10    XX0X  L
0012: 10    XX1X  H
0013: 11    XXX0  L
0014: 11    XXX1  H
0015: 11    XXX0  L
0016: 11    XXX1  H
```

Rys. 45

korzystanie z możliwości układu wirtualnego tym bardziej, że diagnostyka błędów wbudowana w WinCUPL-u nie jest najwyższych lotów. Zdarza się, że komunikat o błędzie wyprowadza projektanta na manowce, sugerując błędy w innych miejscach niż występują w rzeczywistości.

Symulacja w WinCUPL-u

Debugowanie projektu ułatwia wbudowany w WinCUPL-a symulator funkcjonalny, który prezentuje wyniki symulacji w postaci graficznej (na **rys. 44** pokazano widok okna symulatora z wynikami weryfikacji projektu dwuwejściowego multipleksera – jego opis pokazano w EP8/2004 na list. 10).

Do przeprowadzenia symulacji konieczny jest plik wejściowy **.si*, zawierający listę weryfikowanych sygnałów oraz informacje o wektorach wejściowych (pobudzeniach). Na **list. 20** pokazano plik zawierający opis przebiegu symulacji. Jest on dość prosty w analizie:

– Sekcja zaczynająca się od słowa `ORDER` zawiera listę sygnałów wejściowych, wyjściowych i węzłów wewnętrznych, których stany będą monitorowane lub wymuszane.
– Sekcja zaczynająca się od słowa `VECTORS` zawiera informacje o kolejnych (w wierszach) stanach wejść i wyjść. Stany wyjściowe projektant może określić samodzielnie, opisując to, czego się spodziewa, oddać też inicjatywę w ręce symulatora, który określi stany na wyjściach układu.

Tab. 14. Sygnały wykorzystywanych w plikach **.si*

Oznaczenie	Opis	Dotyczy...	Wymuszenie stosowane w graficznym edytorze przebiegów WinCUPL	Symbol
0	Zero logiczne	...wejść	Kliknięcie w dolnej lewej części komórki	–
1	Jedynka logiczna	...wejść	Kliknięcie w górnej lewej części komórki	–
C	Sygnał zegarowy „0-1-0”	...wejść	-[1]	
K	Sygnał zegarowy „1-0-1”	...wejść	-[1]	
X	Stan nieistotny	...wejść	Kliknięcie w środkowej części komórki	
P	Wstępne ładowanie rejestrów	...wejść	Lewy przycisk ALT + kliknięcie myszką w środkowej części komórki	–
H	Stan wysoki na wyjściu testowanego układu	...wyjść	Kliknięcie w dolnej prawej części komórki	–
L	Stan niski na wyjściu testowanego układu	...wyjść	Przycisk SHIFT + kliknięcie w górnej prawej części komórki	–
Z	Stan wysokiej impedancji	...wyjść	Kliknięcie w środkowej części komórki	
*	Stan wyliczany przez symulator	...wyjść	Lewy przycisk CTRL + kliknięcie myszką w środkowej części komórki	–

Interpreter wbudowany w symulator CUPL-a umożliwi m.in. proste formatowanie wyników symulacji generowanych do pliku tekstowego (**.so*, rodzaj raportu z symulacji). Za pomocą znaku `%` można pomiędzy nazwami sygnałów wstawić zadaną liczbę spacji, co ułatwi pogrupowanie sygnałów w taki sposób, aby zwiększyć czytelność raportu. Przykłady plików **.so* bez formatowania i z formatowaniem, uzyskanym poprzez zapisanie linii ze słowem kluczowym `ORDER` w następujący sposób:

```
ORDER: SEL1, SEL0, %4, X0, X1, X2, X3, %2, Y;
```

pokazano na **list. 21**.

Pobudzenia wejść i stany wyjść oznaczane są symbolami, które zebrano w **tab. 14**. Pliki pobudzeń można tworzyć ręcznie za pomocą dowolnego edytora tekstów, można także skorzystać z wbudowanego w WinCUPL-a edytora przebiegów, który znacznie ułatwia (można ręcznie określić stan przypisany określonej liczbie umownych jednostek czasu – **rys. 45**) i przyspiesza zarówno tworzenie samych pobudzeń jak i weryfikację reakcji układu na nie. Twórcy edytora przebiegów zastosowali pomysłowy sposób określania stanów za pomocą myszki: zależy on od miejsca w obrębie komórki wyznaczającej pojedynczy krok na osi czasu, w którym użytkownik kliknie myszką. Przykładowo, kliknięcie w środkowej części komórki powoduje przypisanie stanu wysokiej impedancji, w prawej dolnej części komórki – stanu L, w lewej górnej – stanu „1”, a kliknięcie w środkowej części z jednoczesnym przytrzymaniem klawisza ALT wymusza załadowanie rejestrów predefiniowaną wartością początkową. Niestety, z nieustalonych przyczyn, nie jest możliwe wprowadzenie w ten sposób sygnałów zegarowych i to wbrew zapisom w dokumentacji WinCUPL-a.

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