Układy programowalne, część 7

Siódmą część cyklu poświęcamy przedstawieniu przykładów opisu klasycznych układów synchronicznych, w tym przede wszystkim liczników. Zaczniemy od opisu HDL najważniejszych elementów takich układów: przerzutników.

Automaty – krok 1

Klasyczne układy synchroniczne, często zwane automatami, budowane są na bazie przerzutników, które spełniają w nich rolę pamięci stanu. W zasadzie, poza przerzutnikami RS, wszystkie inne rodzaje przerzutników (T, D, JK, JK-MS) nadają się do spełniania roli pamięci stanu w automatach. W komórkach OLMC układów ispGAL22V10 (taki zastosowano w zestawie AVT-599) znajdują się przerzutniki typu D z wejściami synchronicznego ustawiania i asynchronicznego zerowania.

Na **list. 11...14** znajdują się opisy HDL czterech przerzutników, których praca jest synchronizowana (oprócz przerzutnika RS, który nie wymaga sygnału zegarowego) przez sygnał zegarowy. Użytkownik korzystający z zestawu AVT-599 może wybrać jego źródło (za pomocą jumpera JP4).

Ponieważ – jak już wcześniej wspomniano – w komórkach OLMC układów GAL22V10 znajdują się przerzutniki typu D, opis przedstawiony na list. 11 przygotowano nieco „na wyrost". Zastosowanie w równaniach synchronicznych wejść zerowania (RESET) i ustawiania (PRESET) powoduje jednak, że możliwości funkcjonalne implementowanego przerzutnika są takie same, niezależnie od jego budowy w docelowym układzie PLD. Zwiększamy w ten sposób uniwersalność opisu, który „pasuje" do każdego układu PLD z wbudowanymi jakimikolwiek przerzutnikami D. Z tego samego powodu w opisie nie są wykorzystane dostępne w GAL22V10 sygnały asynchronicznego zerowania Q.AR, NIE_Q.AR oraz synchronicznego ustawiania Q.SP i NIE_Q.SP.

Automaty – krok 2

W zależności od sposobu tworzenia sygnałów wyjściowych, automaty dzielą się na automaty Moore'a (schemat blokowy pokazano na **rys. 33**) i Mealy'ego (schemat blokowy pokazano na **rys. 34**). Różnica pomiędzy nimi polega na tym, że sygnały wyjściowe w automacie Mealy'ego są tworzone w postaci funkcji kombinacyjnej dwóch zmiennych: bitów wyjściowych przerzutników pamiętających aktualny stan automatu (informacja pobierana z wyjść prze-

List. 11. Opis w języku CUPL przerzutnika typu D z synchronicznymi wejściami zerowania i ustawiania

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

/***** Wejscia *****/
PIN 2 = CLK;       /* Wejscie zegarowe */
PIN 6 = DATA;      /* Stan na wejsciu ustala JP2 */
PIN 4 = PRESET;    /* Wejscie synchronicznego ustawiania - JP1*/
PIN 11 = RESET;    /* Wejscie synchronicznego zerowania - B_0 SW1 */

/***** Wyjscia *****/
PIN [17, 18] = [Q, NIE_Q]; /* Wyjscia przerzutnika */

/***** Opis HDL *****/
Q.D = PRESET # (DATA & !RESET);
NIE_Q.D = RESET # (!DATA & !PRESET);
```

List. 12. Opis w języku CUPL przerzutnika typu T z synchronicznymi wejściami zerowania i ustawiania

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

/***** Wejscia *****/
PIN 2 = CLK;       /* Wejscie zegarowe */
PIN 6 = T;         /* Stan na wejsciu T ustala JP2 */
PIN 4 = PRESET;    /* Wejscie synchronicznego ustawiania - JP1*/
PIN 11 = RESET;    /* Wejscie synchronicznego zerowania - B_0 SW1 */

/***** Wyjscia *****/
PIN [17, 18] = [Q, NIE_Q]; /* Wyjscia przerzutnika */

/***** Opis HDL *****/
Q.D = PRESET # (!RESET & !T & Q) # (!RESET & T & NIE_Q);
NIE_Q.D = RESET # (!PRESET & !T & NIE_Q) # (!PRESET & T & Q);
```

List. 13. Opis w języku CUPL przerzutnika typu JK z synchronicznymi wejściami zerowania i ustawiania

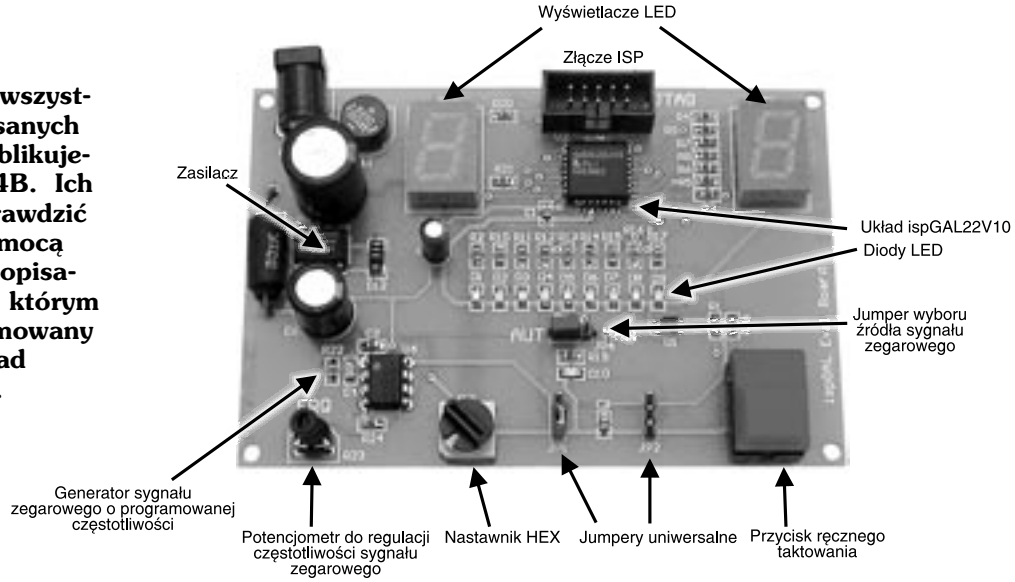
```
Name            jk_ff;
Partno          U1;
Revision        01;
Date            20/05/04;
Designer        PZb;
Company         EP;
Location        brak;
Assembly        brak;
Device          g22v10lcc;

/***** Wejscia *****/
PIN 2 = CLK;       /* Wejscie zegarowe */
PIN 4 = PRESET;    /* Wejscie synchronicznego ustawiania - JP1*/
PIN 6 = RESET;     /* Wejscie synchronicznego zerowania - JP2 */
PIN 7 = J;         /* Wejscie synchronicznego zerowania - B_0 SW1 */
PIN 9 = K;         /* Wejscie synchronicznego zerowania - B_1 SW1 */

/***** Wyjscia *****/
PIN [17, 18] = [Q, NIE_Q]; /* Wyjscia przerzutnika */

/***** Opis HDL *****/
Q.D = PRESET # (J & NIE_Q & !RESET) # (!K & Q & !RESET);
NIE_Q.D = RESET # (!J & NIE_Q & !PRESET) # (K & Q & !PRESET);
```

List. 14. Opis w języku CUPL przerzutnika typu RS (na bramkach NAND)

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

/***** Wejscia *****/
PIN 4 = S;     /* Wejscie ustawiania - JP1*/
PIN 6 = R;     /* Wejscie zerowania - JP2 */

/***** Wyjscia *****/
PIN [17, 18] = [Q, NIE_Q]; /* Wyjscia przerzutnika */

/***** Opis HDL *****/
Q = !S # (R & Q);
NIE_Q = !R # (S & NIE_Q);
```

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.



rzutników spełniających rolę pamięci stanu) i aktualnego stanu wejść automatu. Użytkownik odpowiednio przygotowując opis automatu może wymusić implementację projektu w jednym lub drugim rodzaju automatu. W większości przypadków walory automatów Moore'a są na tyle duże, że implementowane w nim projekty spełniają wymagania typowych aplikacji.

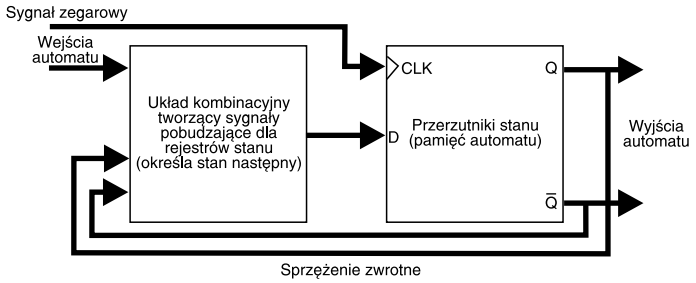
Prezentację opisu automatów zaczniemy od 4-bitowego licznika liczącego w cyklu modulo 16 w dwóch kierunkach. Licznik wyposażono w wejście sychronicznego zerowania. Jego opis HDL przedstawiono na **list. 15**. Ze względu na wygodę zastosowano w nim opis przejść warunkowych pomiędzy kolejnymi stanami automatu, co pozwala na wygodne odwzorowanie standardowego grafu przejść. Ten sam efekt funkcjonalny można uzyskać za pomocą przypisania do wejścia D każdego przerzutnika (Q0.d, Q1.d, Q2.d i Q3.d) wyjściowego funkcji logicznej, ale nie jest to zadanie łatwe do wykonania, na co dowodem może być przedstawione poniżej równanie dla przerzutnika Q3.d:

```
Q3.d =>
  Q0 & Q1 & Q2 & Q3 & !RESET & U_D
  # !Q0 & !Q1 & !Q2 & !Q3 & !RESET & U_D
  # Q0 & Q1 & Q2 & !Q3 & !RESET & !U_D
  # !Q1 & !Q2 & Q3 & !RESET & !U_D
  # Q0 & !Q2 & Q3 & !RESET & U_D
  # !Q0 & Q1 & Q3 & !RESET
  # Q0 & Q1 & !Q2 & Q3 & !RESET & !U_D
  # !Q1 & Q2 & Q3 & !RESET
```

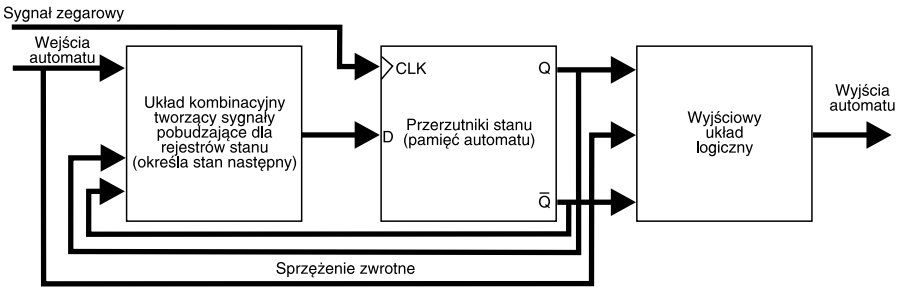
Wykonywanie takich „sztuczek" zaprzecza wygodzie korzystania z języka wysokiego poziomu, więc potraktujmy przedstawioną możliwość jedynie jako przykład możliwości CUPL-a, a nie zalecany sposób projektowania.

O tym, że opis licznika za pomocą równań logicznych nie jest najwygodniejszym sposobem realizacji projektu, nie trzeba nikogo przekonywać. Trzeba jednak przyznać, że także opis kolejnych przejść, jakkolwiek znacznie bardziej czytelny nie jest pozbawiony wady: jest po prostu bardzo długi i w przypadku konieczności wprowadzenia jakiejkolwiek zmiany (np. modyfikacji długości cyklu zliczania lub wprowadzenia dodatkowych sygnałów sterujących), poprawianie tak przygotowanego opisu nie jest wygodne. Sytuację uprości zastosowanie komendy preprocesora `$REPEAT`, jak to pokazano na **list. 16**. Wygodę stosowania zapisów „kompaktowych" zilustrowano na **list. 17**, na którym pokazano opis HDL licznika *modulo 10*, który jest odpowiednikiem funkcjonalnym liczników z list. 15 i 16.

Przedstawione opisy dotyczą pojedynczego licznika, który nie może



Rys. 33



Rys. 34

List. 15. Opis 4-bitowego licznika góra/dół z synchronicznym wejściem zerującym i jawnymi deklaracjami wartości przypisanych kolejnym stanom

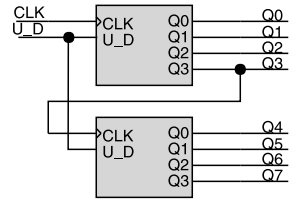
```
Name            cnt_ud_mod16;
Partno          U1;
Revision        01;
Date            20/05/04;
Designer        PZb;
Company         EP;
Location        brak;
Assembly        brak;
Device          g22v10lcc;

/***** Wejscia *****/
PIN 2 = CLK;    /* Wejscie zegarowe */
PIN 4 = RESET;  /* Wejscie asynchronicznego */
                /* zerowania - JP1*/
PIN 6 = U_D;    /* Wybor kierunku zliczania */
                /* - JP2 */

/***** Wyjscia *****/
PIN [17..20] = [Q3..0]; /* Wyjscia licznika */

/***** Deklaracje pomocnicze *****/
field licznik = [Q3..0];
$define S0 ,b'0000
$define S1 ,b'0001
$define S2 ,b'0010
$define S3 ,b'0011
$define S4 ,b'0100
$define S5 ,b'0101
$define S6 ,b'0110
$define S7 ,b'0111
$define S8 ,b'1000
$define S9 ,b'1001
$define S10 ,b'1010
$define S11 ,b'1011
$define S12 ,b'1100
$define S13 ,b'1101
$define S14 ,b'1110
$define S15 ,b'1111

field mode = [RESET,U_D];
up = mode:0;  /* w gore */
down = mode:1; /* w dol */
clear = mode:[2..3]; /* zerowanie */

/***** Opis HDL *****/
sequence licznik {
present S0      if up           next S1;
                if down         next S15;
                if clear        next S0;
present S1      if up           next S2;
                if down         next S0;
                if clear        next S0;
present S2      if up           next S3;
                if down         next S1;
                if clear        next S0;
present S3      if up           next S4;
                if down         next S2;
                if clear        next S0;
present S4      if up           next S5;
                if down         next S3;
                if clear        next S0;
present S5      if up           next S6;
                if down         next S4;
                if clear        next S0;
present S6      if up           next S7;
                if down         next S5;
                if clear        next S0;
present S7      if up           next S8;
                if down         next S6;
                if clear        next S0;
present S8      if up           next S9;
                if down         next S7;
                if clear        next S0;
present S9      if up           next S10;
                if down         next S8;
                if clear        next S0;
present S10     if up           next S11;
                if down         next S9;
                if clear        next S0;
present S11     if up           next S12;
                if down         next S10;
                if clear        next S0;
present S12     if up           next S13;
                if down         next S11;
                if clear        next S0;
present S13     if up           next S14;
                if down         next S12;
                if clear        next S0;
present S14     if up           next S15;
                if down         next S13;
                if clear        next S0;
present S15     if up           next S0;
                if down         next S14;
                if clear        next S0;
}
```

List. 16. Opis 4-bitowego licznika góra/dół z synchronicznym wejściem zerującym i skróconym zapisem deklaracji wartości przypisanych kolejnym stanom

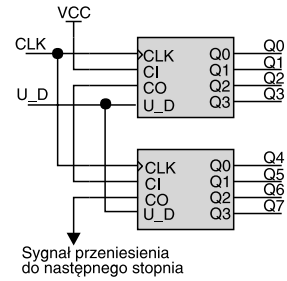
```
Name            cnt_ud_mod16_r;
Partno          U1;
Revision        01;
Date            20/05/04;
Designer        PZb;
Company         EP;
Location        brak;
Assembly        brak;
Device          g22v10lcc;

/***** Wejscia *****/
PIN 2 = CLK;     /* Wejscie zegarowe */
PIN 4 = RESET;   /* Wejscie asynchronicznego */
                 /*zerowania - JP1*/
PIN 6 = U_D;     /* Wybor kierunku zliczania */
                 /*- JP2 */

/***** Wyjscia *****/
PIN [17..20] = [Q3..0]; /* Wyjscia licznika*/

/***** Deklaracje pomocnicze *****/
field licznik = [Q3..0];
$REPEAT i = [0..15]
$DEFINE S{i} {i}
$REPEND

field mode = [RESET,U_D];
up = mode:0;          /* w gore */
down = mode:1;        /* w dol */
clear = mode:[2..3];  /* zerowanie */

/***** Opis HDL *****/
sequence licznik {
PRESENT S0
   IF up NEXT S1;
   IF down NEXT S15;
   IF clear NEXT S0;
$REPEAT i = [1..15]
PRESENT S{i}
   IF up NEXT S{(i+1)%16};
   IF down NEXT S{(i-1)%16};
   IF clear NEXT S0;
$REPEND
}
```

List. 17. Opis 4-bitowego licznika góra/dół liczącego w cyklu modulo 10 z synchronicznym wejściem zerującym i skróconym zapisem deklaracji wartości przypisanych kolejnym stanom

```
Name            cnt_ud_mod10_r;
Partno          U1;
Revision        01;
Date            20/05/04;
Designer        PZb;
Company         EP;
Location        brak;
Assembly        brak;
Device          g22v10lcc;

/***** Wejscia *****/
PIN 2 = CLK;     /*Wejscie zegarowe */
PIN 4 = RESET;   /*Wejscie asynchronicznego*/
                 /*zerowania - JP1*/
PIN 6 = U_D;     /*Wybor kierunku zliczania*/
                 /*- JP2 */

/***** Wyjscia *****/
PIN [17..20] = [Q3..0]; /*Wyjscia licznika*/

/***** Deklaracje pomocnicze *****/
field licznik = [Q3..0];
$REPEAT i = [0..9]
$DEFINE S{i} {i}
$REPEND

field mode = [RESET,U_D];
up = mode:0;            /* w gore */
down = mode:1;          /* w dol */
clear = mode:[2..3];    /* zerowanie */

/***** Opis HDL *****/
sequence licznik {
PRESENT S0
   IF up NEXT S1;
   IF down NEXT S9;
   IF clear NEXT S0;
$REPEAT i = [1..9]
PRESENT S{i}
   IF up NEXT S{(i+1)%10};
   IF down NEXT S{(i-1)%10};
   IF clear NEXT S0;
$REPEND
}
```



Rys. 35



Rys. 36

być łączony w synchroniczne kaskady z innymi, co pozwoliłoby na zwiększenie długości zliczanego słowa. Jest to dość poważna wada przedstawionego rozwiązania, warto więc by było wyposażyć liczniki w wejście i wyjście przeniesienia, które umożliwią ich łączenie w wielobitowe zespoły liczące. Jedyną możliwością zwiększenia długości zliczania jest połączenie ich w sposób pokazany na **rys**. **35**, ale rozwiązanie to ma wadę: naruszana jest synchroniczność licznika, co w przypadku większych częstotliwości taktowania może spowodować niepoprawną jego pracę.

Na **list**. **18** pokazano przykładowy opis 4-bitowego licznika liczącego w cyklu modulo 10, wyposażonego w wejście (CI) i wyjście (CO) przeniesienia (sposób szeregowego łączenia takich liczników pokazano na **rys**. **36**). Opis tego licznika nie różni się zbytnio od wcześniej przedstawionych, należy zwrócić jedynie uwagę na to, że w każdym stanie rozpatrywanych jest więcej warunków, z których jeden (`if others`) zapewnia zatrzymanie się licznika w bieżącym stanie z jego podtrzymaniem. Drugą rzeczą, na którą warto zwrócić uwagę, jest występujące w dwóch miejscach polecenie `out CO`, za pomocą którego „wyprowadzany" jest sygnał przeniesienia.

Automaty – krok 3

Automaty opisywane w języku CUPL można wykorzystać do generowania sygnałów synchronicznych i asynchronicznych, które będą wy-

List. 18. Opis 4-bitowego licznika góra/dół liczącego w cyklu modulo 10 z synchronicznym wejściem zerującym i możliwością łączenia ze sobą liczników w kaskady synchroniczne

```
Name            cnt_ud_mod10_rc;
Partno          U1;
Revision        01;
Date            20/05/04;
Designer        PZb;
Company         EP;
Location        brak;
Assembly        brak;
Device          g22v10lcc;

/***** Wejscia *****/
PIN 2 = CLK;      /*Wejscie zegarowe */
PIN 4 = RESET;    /*Wejscie asynchronicznego*/
                  /*zerowania - JP1*/
PIN 6 = U_D;      /* Wybor kierunku zliczania*/
                  /*- JP2 */
PIN 11 = CI;

/***** Wyjscia *****/
PIN [17..20] = [Q3..0]; /*Wyjscia licznika*/
PIN 26 = CO;

/***** Deklaracje pomocnicze *****/
field licznik = [Q3..0];
$define S0 ,b'0000
$define S1 ,b'0001
$define S2 ,b'0010
$define S3 ,b'0011
$define S4 ,b'0100
$define S5 ,b'0101
$define S6 ,b'0110
$define S7 ,b'0111
$define S8 ,b'1000
$define S9 ,b'1001

field mode = [U_D,CI,RESET];
up = mode:'b'010;        /* w gore */
down = mode:'b'110;      /* w dol */
others = mode:'b'x00;
clear = mode:'b'xx1;
```

List. 18 - cd

```
/***** Opis HDL *****/
sequence licznik {
present S0      if up           next S1;
                if down         next S9;
                if others       next S0;
                if clear        next S0;
present S1      if up           next S2;
                if down         next S0 out CO;
                if others       next S1;
                if clear        next S0;
present S2      if up           next S3;
                if down         next S1;
                if others       next S2;
                if clear        next S0;
present S3      if up           next S4;
                if down         next S2;
                if others       next S3;
                if clear        next S0;
present S4      if up           next S5;
                if down         next S3;
                if others       next S4;
                if clear        next S0;
present S5      if up           next S6;
                if down         next S4;
                if others       next S5;
                if clear        next S0;
present S6      if up           next S7;
                if down         next S5;
                if others       next S6;
                if clear        next S0;
present S7      if up           next S8;
                if down         next S6;
                if others       next S7;
                if clear        next S0;
present S8      if up           next S9 out CO;
                if down         next S7;
                if others       next S8;
                if clear        next S0;
present S9      if up           next S0;
                if down         next S8;
                if others       next S9;
                if clear        next S0;
}
```

stępować wraz z określonymi stanami automatów. Do tego celu służy polecenie `out`, które w przykładzie pokazanym na list. 18 wykorzystano do „wyprowadzenia" z licznika sygnału przeniesienia CO.

Z polecenia `out` można skorzystać na dwa sposoby, uzyskując różne wyniki:

- jeżeli chcemy uzyskać sygnał synchronizowany przebiegiem zegarowym (czyli uzyskiwany na wyjściu przerzutnika taktowanego tym samym sygnałem zegarowym, którym jest taktowany automat), to należy korzystać z następującego zapisu:

```
sequence licznik {
present S0      next S1;
present S1      next S2;
present S2      if A next S3 out CO;
                if !A next S1;
present S3      next S0;
}
```

lub, gdy generowanie sygnału jest bezwarunkowe:

```
equence licznik {
present S0      next S1;
present S1      next S2;
present S2      next S3   out CO;
present S3      next S0;
}
```

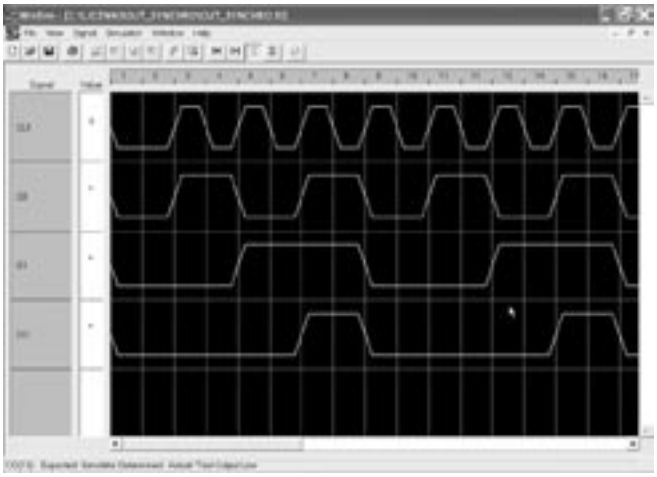
Jakkolwiek takie rozwiązanie jest technicznie eleganckie, należy pamiętać, że generowany sygnał pojawi się na wyjściu opóźniony o jeden takt zegarowy (jeżeli jest generowany w stanie S2, to na wyjściu pojawi się na czas trwania stanu S3). Widać to na **rys. 37** (źródło symulowanego automatu 4-stanowego jest dostępne na CD-EP9/2004B w katalogu *\OUT_synchro* i na stronie internetowej EP w dziale *Download*).

- w przypadku, gdy sygnał wyjściowy ma być wytwarzany w układzie kombinacyjnym (może wtedy zawierać zakłócenia szpilkowe wywołane przez opóźnienia w funktorach logicznych tworzących ten sygnał), zapis w języku CUPL jest następujący:

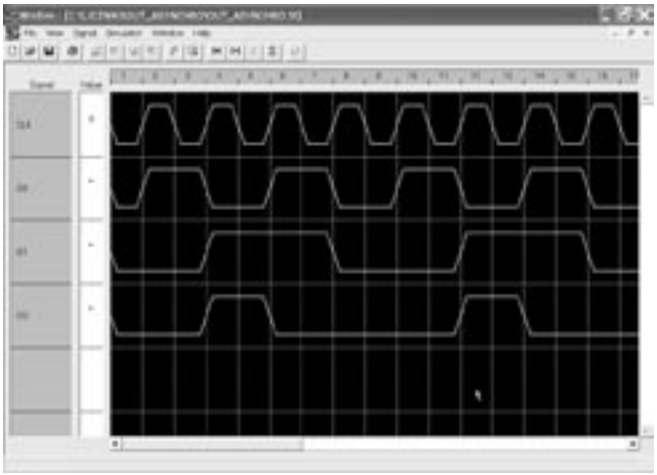
```
sequence licznik {
present S0      next S1;
present S1      next S2;
present S2      next S3;
                out CO;
present S3      next S0;
```

W odróżnieniu od synchronicznego „wyprowadzania" sygnału CO, tym razem pojawia się on dokładnie podczas stanu, do którego go przypisano (**rys. 38**). Nieco więcej zabiegów w tym przypadku wymaga opisanie układu generującego warunkowo sygnał wyjściowy, ponieważ jest on przypisany do danego stanu – jeżeli automat się w nim znajdzie, sygnał wyjściowy na pewno się pojawi. Z tego wynika konieczność wcześniejszego, niż ma to miejsce w układach synchronicznych, rozpatrywania warunków zmiany stanu.

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



Rys. 37



Rys. 38