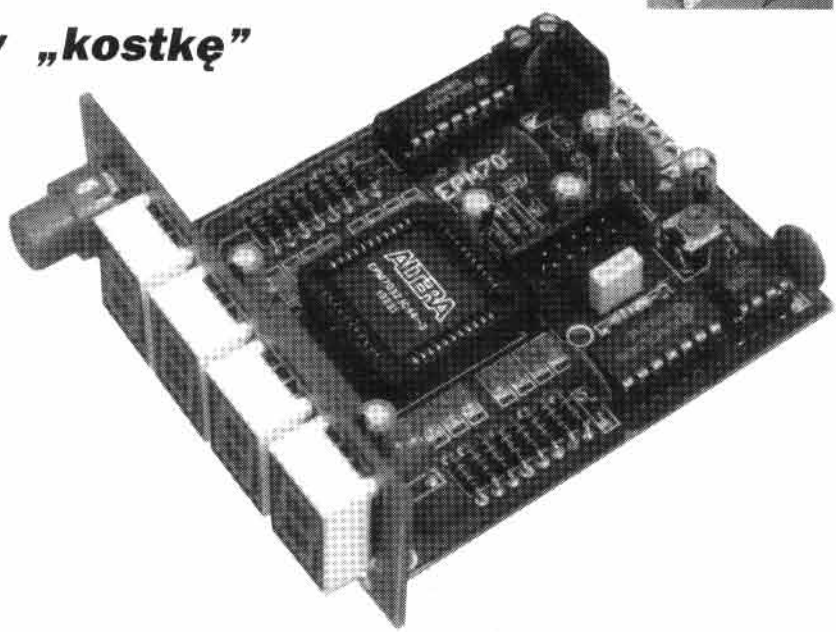
Uniwersalny licznik-timer z układem EPM7032 firmy Altera

Część 2 – budujemy „kostkę"



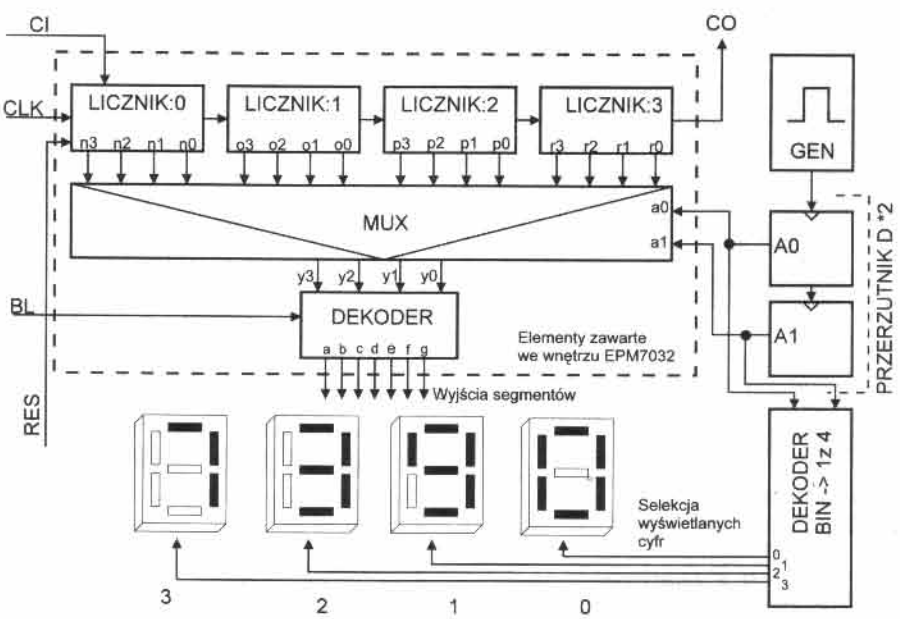
W pierwszej części artykułu przedstawiliśmy oprogramowanie jakim posługiwał się autor podczas projektowania timer'a. W tej części pokażemy krok po kroku w jaki sposób projekt zrealizowano.

Dzięki zastosowaniu bardzo silnego narzędzia w postaci oprogramowania Max+plus II zaprojektowanie nawet mocno rozbudowanych struktur logicznych jest proste.

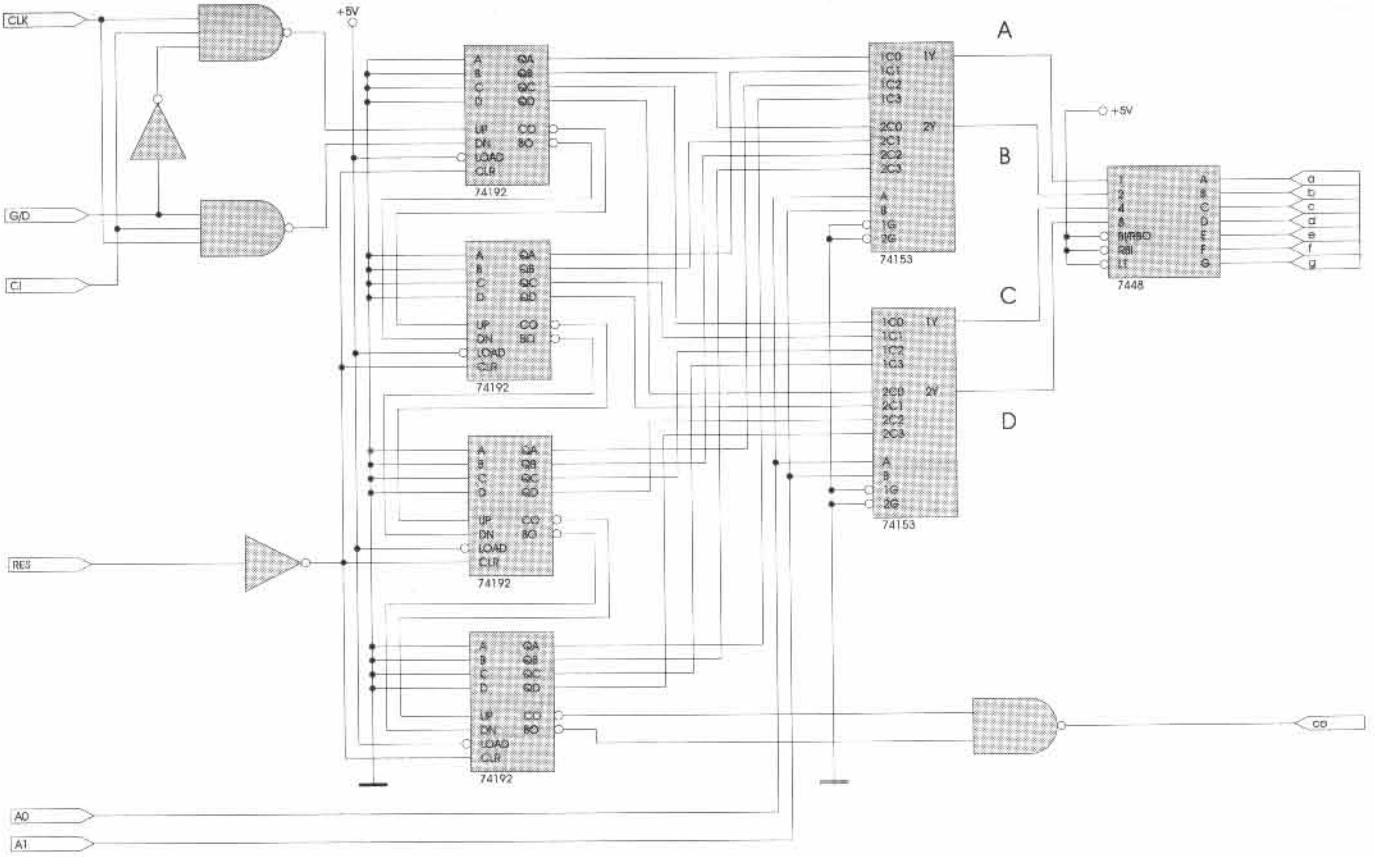
Na początku przypomnimy założenia jakimi kierował się autor podczas projektowania układu:

- budujemy układ uniwersalnego timera - licznika o programowanym kierunku zliczania góra-dół, wyposażonym w wejścia kasujące i strobujące zliczanie. Jako wyposażenie dodatkowe należy wbudować w układ detektor stanu „0000" licznika;
- licznik powinien mieć 4 cyfry (zakres zliczania 0000..9999). Wskazania wyświetlane będą na wskaźnikach 7-segmentowych LED;
- powinna być możliwość łączenia szeregowo kilku układów tego samego typu w celu zwiększenia pojemności licznika;
- wewnątrz najmniejszej kostki rodziny MAX7000 należy „upakować" jak największą ilość elementów timera. Wyraźnym ograniczniem możliwości całkowitej integracji timera w układzie EPM7032 są dwie przyczyny: wyposażony on jest w 32 makrokomórki (co jest zbyt małą ilością dla całego projektu), a bufory wyjściowe mogą pracować z maksymalnym prądem wyjściowym 30mA, co jest zbyt małą wartością dla wysterowania wspólnej elektrody wyświetlacza (konieczne jest zastosowanie drivera o prądzie wyjściowym ok. 50mA);
- projekt powinien być zoptymalizowany pod kątem szybkości pracy. Zalecane jest wykorzystanie maksymalnie możliwości czasowych układu EPM7032-15 (lub inne oznaczenie EPM7032-3), który jest najtańszym członkiem rodziny MAX7000.

Po wstępnej analizie założeń opracowano schemat blokowy struktury układu, co jest bardzo ważnym krokiem podczas opraco-



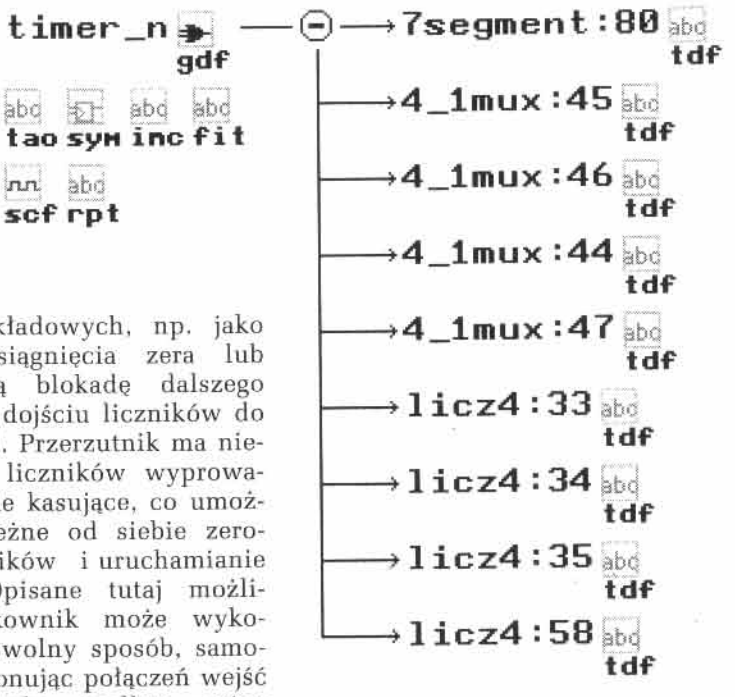
Rys. 1. Schemat blokowy licznika-timera



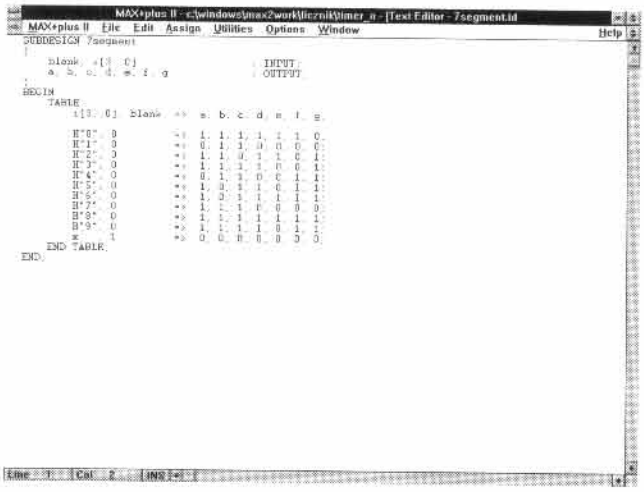
Rys. 2. Schemat ideowy wnętrza projektowanego układu

wywania całej konstrukcji (podobnie zresztą należałoby postąpić podczas projektowania tego samego układu w technice opartej na standardowych układach TTL). Dokładne przemyślenie wszystkich wymogów projektu zaowocowało schematem blokowym, który przedstawiamy na **rys.1**. Linią przerywaną oddzielone zostały elementy projektu, które (z przyczyn podanych powyżej) będą się znajdowały poza naszą „kostką". Jak widać, wyświetlanie wskazań będzie się odbywało sekwencyjnie, dzięki czemu niezbędny będzie tylko jeden dekoder kodu BCD na kod wyświetlacza 7-segmentowego. Multipleksery sterowane z zewnętrznego dwubitowego licznika będą przyłączać kolejno wyjścia liczników do wejść dekodera. Jednocześnie dekoder cyfry (zewnętrzny) będzie kolejno umożliwiał zapalenie jednego tylko wyświetlacza, którego numer (od 0..3) będzie odpowiadał numerowi licznika dołączonego do wejść dekodera. Jako element podnoszący funkcjonalność układu w jego wnętrze wbudowany został dodatkowy przerzutnik typu D z wykonanym na bramkach OR i AND detektorem stanu „0000" licznika. Wyjście !Q przerzutnika wyprowadzono na zewnątrz, co pozwala wykorzystać je w różnych konfiguracjach układowych, np. jako wskaźnik osiągnięcia zera lub automatyczną blokadę dalszego zliczania po dojściu liczników do stanu „0000". Przerzutnik ma niezależnie od liczników wyprowadzone wejście kasujące, co umożliwia niezależne od siebie zerowanie liczników i uruchamianie zliczania. Opisane tutaj możliwości użytkownik może wykorzystać w dowolny sposób, samodzielnie dokonując połączeń wejść i wyjść układu według opisu przedstawionego dalej.

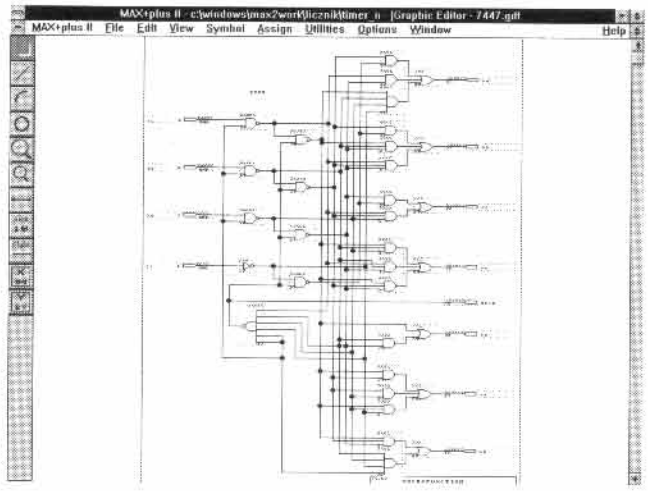
Od schematu blokowego możemy wprost przejść do schematu ideowego wykonanego w oparciu o typowe kostki TTL. Na **rys.2** przedstawiamy taki schemat wy-



Rys. 3. Struktura hierarchii projektu



Rys. 4. Projekt dekodera w języku AHDL



Rys. 5. Logiczna struktura dekodera

konany w edytorze graficznym programu OrCAD SDT. Ma on służyć tylko do ułatwienia porównania rozmiarów (fizycznych i strukturalnych) układu wykonanego standardowo i z wykorzystaniem techniki PLD. Nie wykorzystane wejścia wpisu równoległego do liczników 74192 w czasie analizy schematu można pominąć - kompilator w MaxPlus+II poddałby te wejścia minimalizacji ponieważ nie są one wykorzystywane w czasie pracy układu. Łatwo sobie uzmysłowić trudności z zaprojektowaniem płytki drukowanej pod ten stosunkowo rozbudowany układ. Mając taki punkt odniesienia przechodzimy do projektowania pod opieką Max+Plus II.

```
SUBDESIGN licz4

(
  clk, res, ci, u/d    :INPUT;
  q[3..0]              :OUTPUT;
  co                   :OUTPUT;
)

VARIABLE
  ss: MACHINE OF BITS (q[3..0])
      WITH STATES ( s0 = B"0000",
                    s1 = B"0001",
                    s2 = B"0010",
                    s3 = B"0011",
                    s4 = B"0100",
                    s5 = B"0101",
                    s6 = B"0110",
                    s7 = B"0111",
                    s8 = B"1000",
                    s9 = B"1001");

BEGIN
  ss.clk = clk;
  ss.reset = res;

  TABLE
  ss,   ci,   u/d   =>   ss,   co;
  s0,   1,    0          =>   s9,   1;
  s0,   1,    1          =>   s1,   0;
  s0,   0,    x          =>   s0,   0;
  s1,   1,    0          =>   s0,   0;
  s1,   1,    1          =>   s2,   0;
  s1,   0,    x          =>   s1,   0;
  s2,   1,    0          =>   s1,   0;
  s2,   1,    1          =>   s3,   0;
  s2,   0,    x          =>   s2,   0;
  s3,   1,    0          =>   s2,   0;
  s3,   1,    1          =>   s4,   0;
  s3,   0,    x          =>   s3,   0;
  s4,   1,    0          =>   s3,   0;
  s4,   1,    1          =>   s5,   0;
  s4,   0,    x          =>   s4,   0;
  s5,   1,    0          =>   s4,   0;
  s5,   1,    1          =>   s6,   0;
  s5,   0,    x          =>   s5,   0;
  s6,   1,    0          =>   s5,   0;
  s6,   1,    1          =>   s7,   0;
  s6,   0,    x          =>   s6,   0;
  s7,   1,    0          =>   s6,   0;
  s7,   1,    1          =>   s8,   0;
  s7,   0,    x          =>   s7,   0;
  s8,   1,    0          =>   s7,   0;
  s8,   1,    1          =>   s9,   0;
  s8,   0,    x          =>   s8,   0;
  s9,   1,    0          =>   s8,   0;
  s9,   1,    1          =>   s0,   1;
  s9,   0,    x          =>   s9,   0;
  END TABLE;

END;
```

Listing 1. Program w języku AHDL opisujący pracę licznika

```
SUBDESIGN 7segment
(
  blank, i[3..0]        : INPUT;
  a, b, c, d, e, f, g   : OUTPUT;
)
BEGIN
  TABLE
    i[3..0], blank => a, b, c, d, e, f, g;

    H"0", 0             => 1, 1, 1, 1, 1, 1, 0;
    H"1", 0             => 0, 1, 1, 0, 0, 0, 0;
    H"2", 0             => 1, 1, 0, 1, 1, 0, 1;
    H"3", 0             => 1, 1, 1, 1, 0, 0, 1;
    H"4", 0             => 0, 1, 1, 0, 0, 1, 1;
    H"5", 0             => 1, 0, 1, 1, 0, 1, 1;
    H"6", 0             => 1, 0, 1, 1, 1, 1, 1;
    H"7", 0             => 1, 1, 1, 0, 0, 0, 0;
    H"8", 0             => 1, 1, 1, 1, 1, 1, 1;
    H"9", 0             => 1, 1, 1, 1, 0, 1, 1;
    x   , 1             => 0, 0, 0, 0, 0, 0, 0;
  END TABLE;
END;
```

Listing 2. Program w języku AHDL opisujący dekoder BCD -> 7 segm

```
SUBDESIGN 4_1mux

(w3, w2, w1, w0       :INPUT;
 a1, a0               :INPUT;
 y                    :OUTPUT;
)

BEGIN
        y = a1 & a0 & w3 #
                      a1 & !a0 & w2 #
                      !a1 & a0 & w1 #
                      !a1 & !a0 & w0;
END;
```

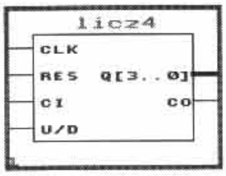
Listing 3. Program w języku AHDL opisujący multiplekser 4-bitowy

Realizowany przez nas projekt będzie oparty na strukturze hierarchicznej dwupoziomowej. Strasznie zabrzmiało? Naprawdę jest to proste: projekt będzie polegał na narysowaniu schematu ideowego układu (jest to wyższy stopień hierarchii) z wykorzystaniem bloczków (tzn. liczników, dekodera, multiplekserów itd.) opracowanych wcześniej przy pomocy edytora tekstowego w języku AHDL. Te właśnie bloczki stanowią niższy stopień w hierarchii projektu (**rys. 3**) i wszystkie wymienione są na tym samym poziomie. Przypomina to budowę dowolnego innego urządzenia składającego się z wielu części, np. samochodu. Każdy z jego elementów wykonuje jakiś zakład (niższy stopień hierarchii), a całość składana jest w montowni (wyższy stopień hierachii) z gotowych kawałków.

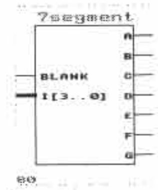
Zaczynamy od klocków

Na schemacie blokowym (**rys.1**) łatwo wyróżnić bloki funkcjonalne, które będą nam niezbędne do zrealizowania projektu. Są to:

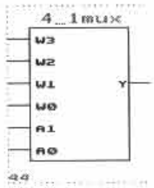
- czterobitowy licznik BCD, wyposażony w wejście zegarowe (oznaczymy je jako CLK), wejście kasujące (ozn. RES), wejście strobujące zliczanie (ze względu na możliwość pracy tego wejścia jako odbiorcy sygnału przeniesienia z poprzedniego stopnia w przypadku połączenia szeregowego kilku liczników sygnał ten nazwano CI, z ang. Carry Input), wejście ustalające kierunek zliczania (ozn. U/D) oraz wyjścia zliczające Q[3..0] i wyjście gene-



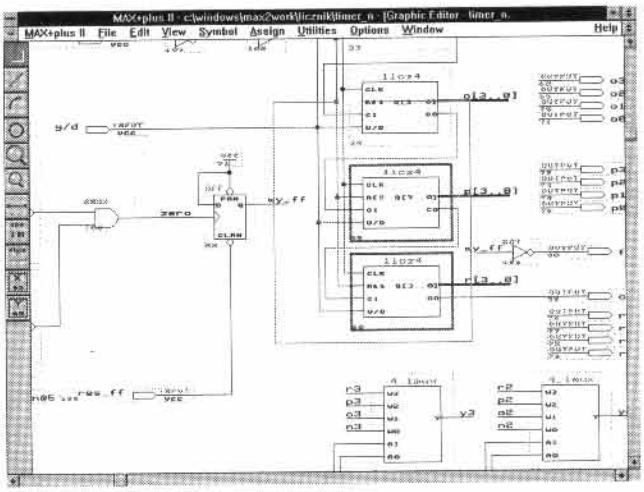
Rys. 6.



Rys. 7.



Rys. 8.



Rys. 9. Edytor schematów systemu MAX+plus II

racji przeniesienia do następnego stopnia (ozn. CO);
- multiplekser 4->1 wyposażony w wejścia adresowe (ozn. A0, A1), cztery wejścia danych (ozn. W[3..0]) oraz wyjście wybranej danej (ozn. Y);
- dekoder do sterowania wyświetlaczem 7-segmentowym. Przystosowany on jest do zasilania wyświetlacza o wspólnej katodzie, a jako wyposażenie dodatkowe wyposażono go w wejście oznaczone jako BLANK, które pozwala na wyłączenie wyświetlania poprzez wymuszenie stanu „0" na wszystkich wyjściach segmentowych A..G. Sygnał ten podłączono do jednego z pinów układu EPM7032.

Funkcjonalne odpowiedniki trzech wymienionych powyżej elementów są dostarczane w ramach standardowych bibliotek przez Alterę. Możliwe jest więc po prostu „sięgnięcie" do gotowych bibliotek, narysowanie schematu i skompilowanie tak przygotowanego projektu, ale.. Ze względu na wzorowanie bibliotek na serii układów TTL nie udało się dobrać dokładnie takich liczników i dekodera o jakie nam chodziło w projekcie. Wykorzystanie gotowych elementów (liczniki 74192, dekoder 7448 i multipleksery 74153) spowodowało pewne „rozrośnięcie" się projektu uniemożliwiając jego pełne upakowanie w kostce o 32 makrokomórkach. Nie jest to znacząca wada systemu - budując urządzenia w dyskretnej technice TTL też spotykamy się często z problemem niepełnego dopasowania funkcjonalności elementów do naszych potrzeb. Chcąc niechcąc trzeba było wykonać te elementy samodzielnie. Możliwości było kilka - najprostsze z nich to narysowanie przy pomocy bramek logicznych i przerzutników schematów pożądanych przez nas elementów lub opisanie ich tekstowo za pomocą języka AHDL (ang. Altera Hardware Description Language). Pozornie łatwiejsze, pierwsze rozwiązanie, szybko przekonało autora do sięgnięcia po edytor tekstowy. Dla porówania stopnia złożoności obydwu metod na **rys.4** przedstawiono opis dekodera tekstem, a na **rys.5** przy pomocy edytora logicznego. Obydwa zapisy dają identycznie działające układy! Jasny jest więc powód dla którego jako podstawę do tworzenia nowych elementów wybrano AHDL. List. 1 przedstawia zapis który po kompilacji pozwala stworzyć element nazwany „licz4". Symbol graficzny licznika przedstawia **rys.6**. Podobnie list.2 wraz z **rys.7** przedstawiają dekoder wyświetlacza, a list.3 i **rys.8** multiplekser wykorzystany w projekcie. Graficzne symbole wszystkich dołączanych nowych elementów automatycznie tworzy kompilator. Tak więc dzięki elastyczności systemu projektowego możemy dowolnie rozbudowywać biblioteki dostarczone przez producenta, wciągając do nich elementy tworzone dla różnych projektów.

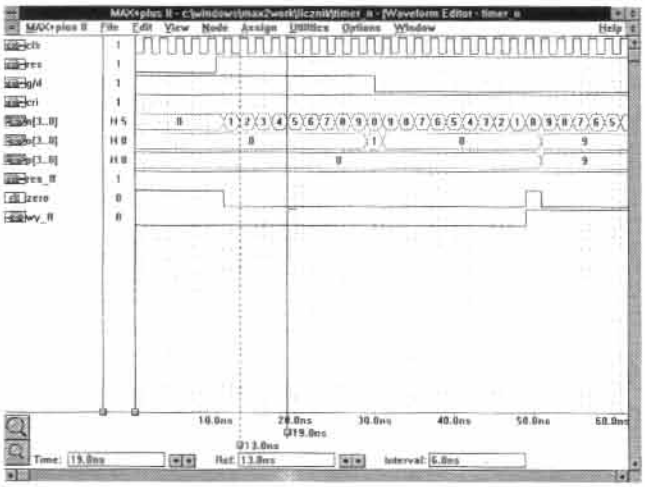
Pozostałe elementy, tzn. bramki logiczne (AND, NOR, NOT) i przerzutnik D są tzw. prymitywami systemu, co oznacza że są to elementy znajdujące się w strukturze układów MAX7000, stanowiące podstawę do syntezy bardziej złożonych elementów (liczników, dekoderów itp.).

Mamy już klocki, budujemy wnętrze układu...

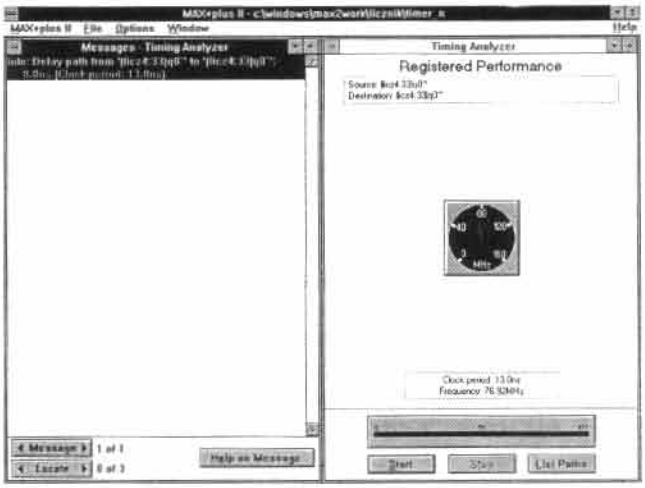
Mamy już wszystkie niezbędne elementy, możemy więc rozpocząć rysowanie schematu układu w oparciu o wcześniej przygotowany schemat blokowy. Na **rys.9** przedstawiamy planszę edytora graficznego systemu Max+Plus II. Rysowanie schematu jest niezwykle proste. Osoby mające minimalne choćby doświadczenie w posługiwaniu się programami graficznymi pracującymi pod Windows doskonale będą sobie potrafiły dać radę. Po stworzeniu przez kompilator zaprojektowanych przez nas elementów są one automatycznie dołączane do istnieją-

```
Device-Specific Information:
timer_n
c:\windows\max2work\licznik\timer_n.rpt
***** Logic for device 'timer_n' compiled without errors.

Device: EPM7032JC44-15
Turbo: ON
Security: OFF

               R
               E
               S  r  f
               E  e  f
               R  s                             a  a
               V  _  o  V  g  r  G  c  G  d  d
               E  f  u  C  /  e  N  l  N  r  r
               D  f  t  C  d  s  D  k  D  1  0
            -----------------------------------------------
          /    6  5  4  3  2  1 44 43 42 41 40   |
     r3 |  7                                  39 | n3
     r2 |  8                                  38 | n2
     r1 |  9                                  37 | n1
    GND | 10                                  36 | n0
     r0 | 11                                  35 | VCC
  blank | 12         EPM7032JC44-15           34 | o3
     p3 | 13                                  33 | o2
     p2 | 14                                  32 | o1
    VCC | 15                                  31 | o0
     p1 | 16                                  30 | GND
     p0 | 17                                  29 | RESERVED
        |_ 18 19 20 21 22 23 24 25 26 27 28  _|
            -----------------------------------------------
               a  b  c  d  G  V  e  f  g  c  c
                           N  C           r  r
                           D  C           i  o

N.C. = Not Connected.
VCC = Dedicated power pin,
which MUST be connected to VCC.
GND = Dedicated ground pin or unused dedicated input,
which MUST be connected to GND.
RESERVED = Unused I/O pin, which MUST be left unconnected.
```

Listing 4. Widok wyprowadzeń zaprojektowanego układu (fragment raportu generowanego po kompilacji)



Rys. 10. Wyniki analizy funkcjonalnej projektu



Rys. 11. Wyniki analizy czasowej projektu

cych bibliotek. Tak więc możemy korzystać z nich na takich samych zasadach jak z pozostałych elementów - wybieramy je z menu wyświetlanego po dwukrotnym wciśnięciu przycisku myszy w wybranym miejscu „kartki", a połączenia pomiędzy wejściami i wyjściami elementów dokonuje się poprzez wybranie punktu początkowego połączenia i ciągnięcie myszy z wciśniętym lewym przyciskiem do punktu końcowego. Ostatnim etapem rysowania schematu jest podłączenie końcówek wejść i wyjść do układu. Określają one fizyczne styki (wyprowadzenia) układu scalonego. Możliwe jest narzucenie numeracji wyprowadzeń, co może znacznie uprościć wykonanie płytki drukowanej.

Tak przygotowany schemat poddajemy dwukrotnej kompilacji, co pozwoli wykryć wszystkie nieprawidłowości i błędy popełnione podczas rysowania schematu. Pierwsza kompilacja powinna się odbyć z włączoną opcją tworzenia pliku dla symulatora funkcjonalnego. Jest to nieco uproszczona kompilacja i z tego powodu nie jest tworzony plik *.sof dla programatora. Kompilator przygotowuje tylko podstawowe pliki dla analizatora logicznego - na **rys.10** widać okno symulatora po analizie układu timera z uwzględnionymi stanami wszystkich wejść i wyjść zaprojektowanego układu. Symulacja funkcjonalna wykonana przez Max+Plus II pozwala na zdiagnozowanie poprawności pracy układu przed zaprogramowaniem fizycznej kostki. Wydłuża to nieco proces projektowania ale znakomicie upraszcza uruchamianie urządzenia.

Druga kompilacja trwa nieco dłużej - tworzone są pliki dla programatora, bardzo precyzyjny, kilkunastostronicowy raport z całego procesu kompilacji oraz pliki umożliwiające analizatorowi czasowemu wykonanie obliczeń określających maksymalną częstotliwość zegarową dla układu synchronicznego i czasy propagacji sygnałów od wejść do wyjść dla układu kombinacyjnego. W przypadku znacznego ograniczenia maksymalnej częstotliwości pracy z powodu złego (z punktu widzenia szybkich sygnałów) rozkładu wnętrza układu analizator czasowy wskazuje połączenia wprowadzające te ograniczenia, co pozwala na modyfikację rozkładu (w oryginale: fittingiu) wnętrza układu. Jest to doskonałe narzędzie dla projektantów układów pracujących w krytycznych czasowo i częstotliwościowo warunkach. **Rys. 11** przedstawia widok ekranu analizatora czasowego z wyliczoną maksymalną częstotliwością pracy dla naszego projektu.

Obydwa wymienione procesy analizy można przeprowadzić nie tylko dla całego projektu ale także dla każdego elementu z osobna. Ma to dość duże znaczenie w przypadku wprowadzania nowych elementów do bibliotek ponieważ pozwala na wychwycenie błędów na wstępnym etapie realizowania projektu. Sposób przeprowadzenia tych analiz jest identyczny jak w przypadku całego projektu, a nie opisaliśmy ich szczegółowo aby nie zamazywać ogólnego obrazu metodyki realizowania projektu.

Mamy już gotowy i skompilowany projekt (cała wiedza o nim zapisana jest do pliku o rozszerzeniu *.sof) - musimy posłużyć się teraz niestety programatorem (układ EPM7032 potrafi zaprogramować większość programatorów uniwersalnych dobrych firm) i mamy w ręku samodzielnie wykonany miniASIC! Pomimo ogromnej prostoty zrealizowanego projektu i pozornej błahości zastosowania nie należy lekceważyć przedstawionych metod projektowania. Już wkrótce przedstawimy Czytelnikom „prawdziwy" ASIC wykonany w oparciu o znacznie większy układ z rodziny MAX7000 - EPM7128, którego proces projektowania przebiegał dokładnie w taki sam sposób jaki teraz przedstawiamy.

W trzeciej części artykułu opiszemy konstrukcję timera wykorzystującego zaprojektowany przez nas układ.

Piotr Zbysiński, AVT