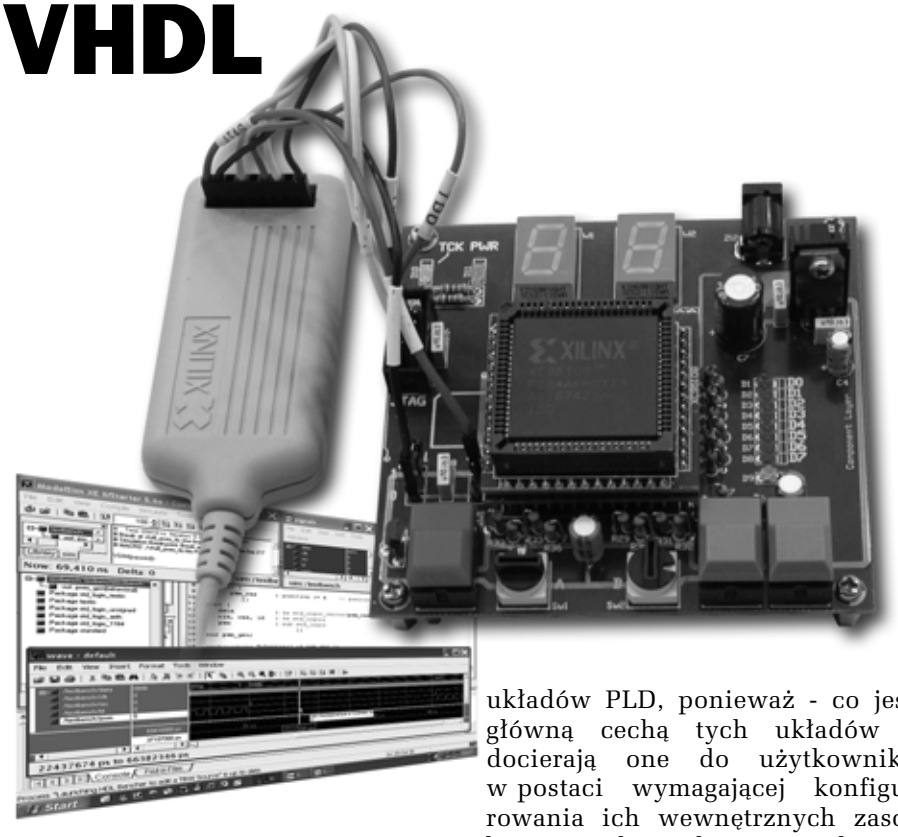
Programowany generator PWM w VHDL

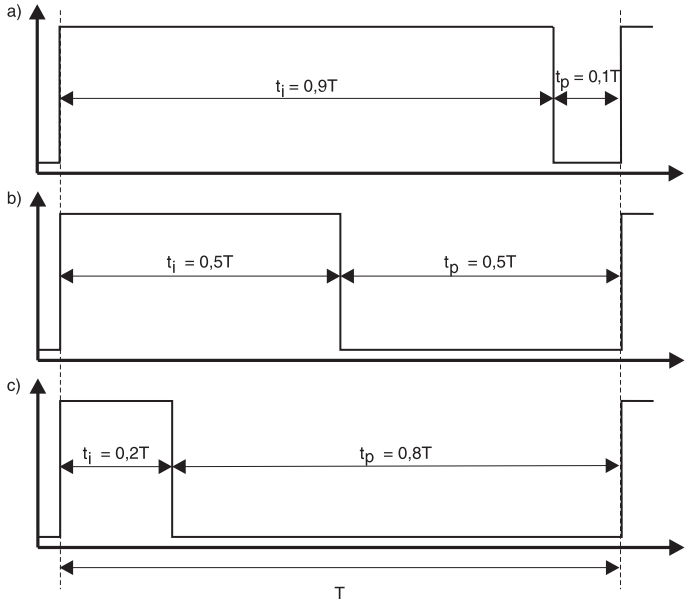


W sprzętowe generatory przebiegów prostokątnych o zmiennym współczynniku wypełnienia (PWM - Pulse Width Modulation) są wyposażone praktycznie wszystkie współczesne mikrokontrolery. Nie dzieje się tak bez przyczyny - za ich pomocą można zmieniać m.in. obroty silników elektrycznych, jasność świecenia żarówek, można je także wykorzystać jako przetworniki C/A o całkiem niezłych parametrach. Dzięki zastosowaniu uniwersalnego języka opisu sprzętu, prezentowany w artykule projekt można traktować jak klasyczny blok IP (Intellectual Property core).

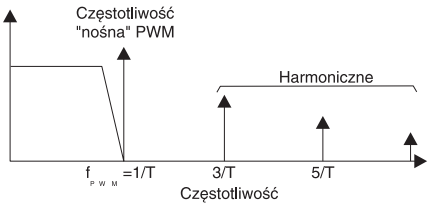
***Rekomendacje**: polecamy projektantom układów cyfrowych, którzy chcą efektywnie wykorzystywać możliwości nowoczesnych układów programowalnych.*

Niewątpliwie użytkownicy mikrokontrolerów mają wygodne życie, ponieważ generatory PWM, podobnie do wielu innych przydatnych modułów i interfejsów dostają gotowe w cenie mikrokontrolera „z półki". Nieco gorzej wygląda sytuacja użytkowników układów PLD, ponieważ - co jest główną cechą tych układów - docierają one do użytkownika w postaci wymagającej konfigurowania ich wewnętrznych zasobów w celu realizowania dowolnych zadanych przez użytkownika funkcji.

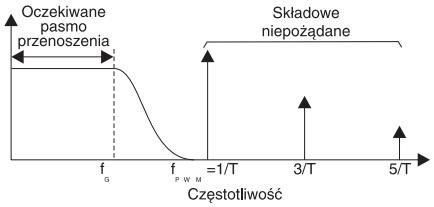
Czy oznacza to, że fani PLD nie mogą wyposażać swoich projektów w UART-y, interfejsy I^2C czy generatory PWM? Oczywiście nie, o czym postaram się przekonać Czytelników w najbliższych



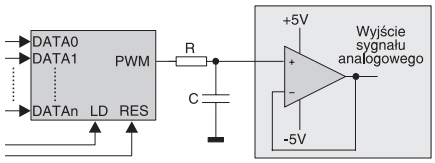
Rys. 1. Trzy przebiegi o współczynniku wypełnienia: a) 90%, b) 50%, c) 20%



Rys. 2. Charakterystyka widmowa generatora sygnału prostokątnego



Rys. 3. Charakterystyka widmowa sygnału na wyjściu generatora PWM po zastosowaniu filtru dolnoprzepustowego



Rys. 4. Schemat filtru dolnoprzepustowego RC pierwszego rzędu (wtórnik napięciowy zalecany, lecz niekonieczny)

numerach EP. Zaczynamy od modułu, który cieszy się dużym zainteresowaniem wśród piszących do mnie Czytelników: programowanym generatorem PWM.

Co to jest PWM?

Modulacja PWM polega na modyfikowaniu szerokości wybranej części („0" lub „1") przebiegu prostokątnego, którego częstotliwość jest stała. Oznacza to, że w zależności od wartości współczynnika wypełnienia, czasy trwania „impulsów" t_i (zazwyczaj „1") i „przerw" pomiędzy nimi t_p (zazwyczaj „0") zmieniają się, przy czym spełniana jest zależność $t_i + t_p = T = const.$ (czyli częstotliwość generowanego przebiegu jest stała). Wartość współczynnika wypełnienia jest podawana zazwyczaj w procentach. Określa się ją wzorem: $\alpha = (t_i/T)*100$ [%]. Na **rys. 1** pokazano trzy przykładowe fragmenty przebiegów o różnych współczynnikach wypełnienia.

Jak wspomniałem na wstępie artykułu, generator przebiegu PWM można zastosować m.in. jako przetwornik C/A. Jak to jest możliwe, jeśli na wyjściu generatora występuje przebieg cyfrowy? Otóż, średnia wartość napięcia na wyjściu generatora przebiegu PWM jest proporcjonalna do wartości współczynnika wypełnienia generowanego przebiegu. Bezpośrednie wykorzystanie sygnału występującego na wyjściu generatora PWM, na przykład do odtwarzania sygnałów akustycznych, nie jest możliwe z powodu wysokiego poziomu składowych harmonicznych w jego widmie (jak pokazano na **rys. 2**). Zmniejszenie ich poziomu wymaga zastosowania prostego filtru dolnoprzepustowego, którego częstotliwość graniczna f_G (-3dB) będzie mniejsza od częstotliwości sygnału PWM ($f_{PWM}=1/T$, **rys. 3**). Schemat typowego filtru RC pierwszego rzędu pokazano na **rys. 4**. Ze względu na jego prostotę (i - niestety - stosunkowo niewielką skuteczność tłumienia niepożądanych harmonicznych), zalecane jest kilkukrotne zwiększenie „odstępu" pomiędzy częstotliwością graniczną filtru dolnoprzepustowego a częstotliwością f_{PWM} tak, żeby spełnić warunek $f_{PWM}=n*f_G$, przy czym *n* powinno mieć wartość co najmniej 3, a w większości przypadków (zwłaszcza podczas odtwarzania sygnałów audio) nawet 5. Wartości parametrów elementów RC filtru dolnoprzepustowego można obliczyć korzystając ze wzoru $R = 1/(2*\pi*f_G*C)$.

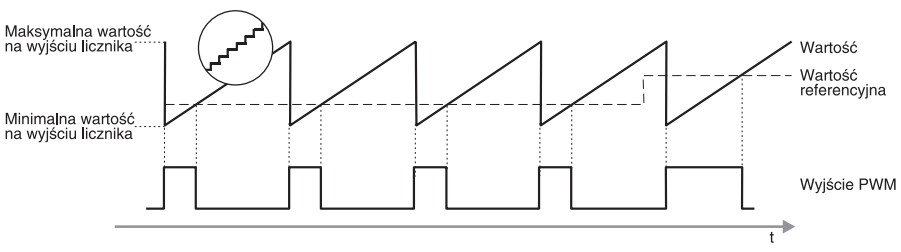
Źródła w Internecie

Pliki źródłowe projektu prezentowanego w artykule są dostępne na stronie internetowej EP w dziale *Download>Dokumentacje*.

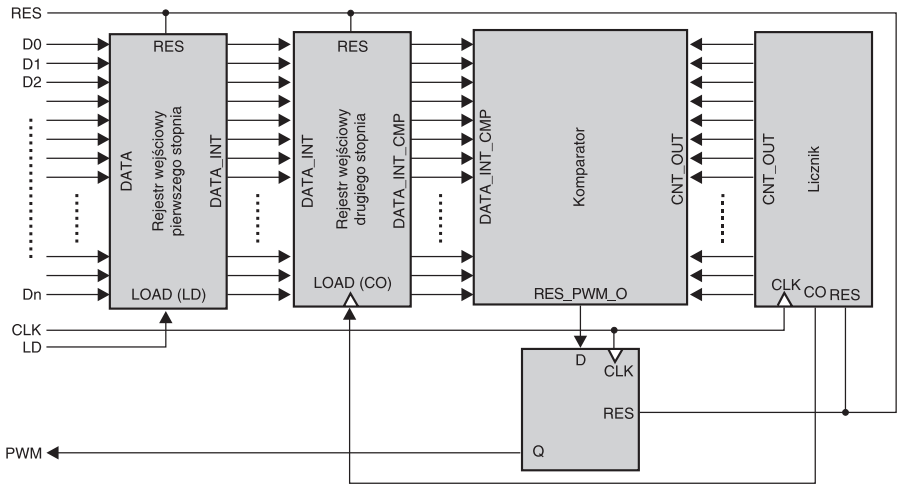
Jak to zrobić w VHDL-u?

Programowany generator PWM można wykonać na wiele różnych sposobów. Prezentowany w artykule projekt można zakwalifikować jako „klasyczny" - jego zasadę działania zilustrowano na **rys. 5**. Pokazany na tym rysunku przebieg schodkowy symbolizuje zmianę stanów na wyjściu licznika binarnego zliczającego w cyklu *modulo* 2^n. Liczba zliczonych przez licznik impulsów jest porównywana przez komparator z liczbą referencyjną podaną przez użytkownika. Jest ona traktowana jako wartość określająca czas trwania (liczony w cyklach zegarowych) stanu „1" na wyjściu generatora PWM. Odliczanie czasu trwania „1" na wyjściu zaczyna się zawsze przy stanie licznika „0". Wartość parametru *n* określa rozdzielczość generowanego przebiegu, czyli - inaczej mówiąc - liczbę możliwych do ustawienia wartości współczynnika wypełnienia. Przykładowo, gdy n = 2, generowany przebieg PWM może mieć współczynnik wypełnienia o jednej z wartości: $^0/_4$, $^1/_4$, $^2/_4$ i $^3/_4$. Zwiększenie długości licznika o jeden bit (czyli n = 3) zwiększa dwukrotnie liczbę możliwych nastaw: $^0/_8$, $^1/_8$, $^2/_8$, $^3/_8$, $^4/_8$, $^5/_8$, $^6/_8$ i $^7/_8$. Jak widać, zwiększanie długości licznika powoduje zwiększenie rozdzielczości programowania współczynnika wypełnienia przebiegu, przy czym najczęściej są stosowane 8...10-bitowe generatory PWM.

Schemat blokowy generatora PWM, którego opis w języku VHDL zaprezentujemy w dalszej części artykułu, pokazano na **rys. 6**. Jest on nieco bardziej rozbudowany niż wynika z dotychczasowego opisu, co zostało spowodowane dwoma czynnikami:



Rys. 5. Ilustracja zasady działania generatora PWM



Rys. 6. Schemat blokowy generatora PWM

- Ze względu na chęć zapewnienia wysokiej jakości sygnału PWM (brak impulsów *glitch* na tym wyjściu), jest on generowany synchronicznie, co wymagało zastosowania dodatkowego przerzutnika D.
- Aby ułatwić współpracę generatora z systemem mikroprocesorowym lub dowolnym innym urządzeniem, na wejściu danych (tam, gdzie jest wpisywana wartość odniesienia) zastosowano dwustopniowy rejestr *latch*. Dzięki temu zmiana wartości odniesienia powoduje zmianę współczynnika wypełnienia dopiero po zakończeniu bieżącego cyklu odliczania, co z kolei powoduje, że nie występują impulsy *glitch* na wyjściu PWM (jak ma to na przykład miejsce w niektórych mikrokontrolerach AVR).

Na **list. 1** przedstawiono opis w języku VHDL układu, którego schemat blokowy pokazano na rys. 6. Rejestr pierwszego stopnia jest zapisywany sygnałem logicznym „1" podawanym na wejście *ld* (wejście zewnętrznego sygnału zapisującego) - stąd warunek w jego opisie `elsif ld = '1' then....` Rejestr drugiego stopnia jest zapisywany narastającym zboczem sygnału na wejściu *co*, co zapisano jako `elsif rising_edge(co) then....` Sygnał *co* jest wytwarzany przez synchroniczny komparator zawsze, gdy wszystkie wyjścia licznika przyjmują stany „1" (następuje przepełnienie licznika). Dzięki temu przepisanie z rejestru pierwszego stopnia na wejścia komparatora nowej wartości referencyjnej następuje zawsze po zakończeniu pełnego cyklu zliczania wynikającego z długości cyklu licznika.

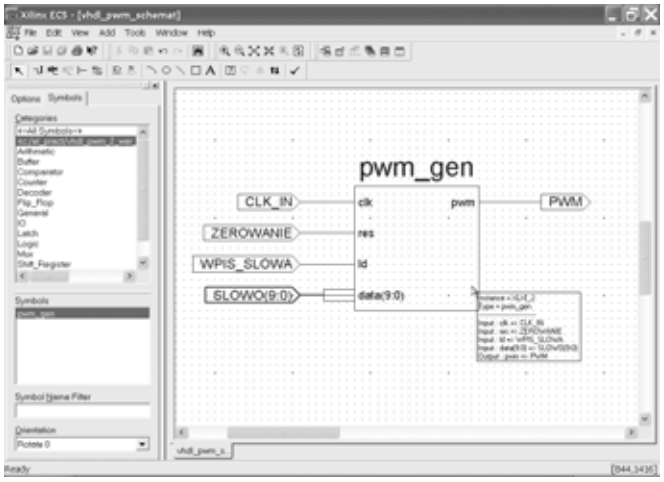
Opis licznika zastosowanego w generatorze jest niezwykle prosty, a to dzięki możliwości przeciążenia operatora „+" (zliczanie kolejnych impulsów zapisano jako cnt_out <= cnt_out + 1;). Wymagało to zastosowania biblioteki STD_LOGIC_ARITH z pakietu IEEE, która jest dostarczana z większością współczesnych systemów projektowych (za wyjątkiem MAX+Plus II). Opis komparatora wykrywającego przekroczenie wartości referencyjnej jest wydzielonym procesem, w wyniku syntezy którego powstaje układ kombinacyjny wykrywający warunek *data_int_cmp* ≥ *cnt_out*, gdzie *data_int_cmp* - to wartość referencyjna, a *cnt_out* - bieżący stan

Narzędzia za darmo

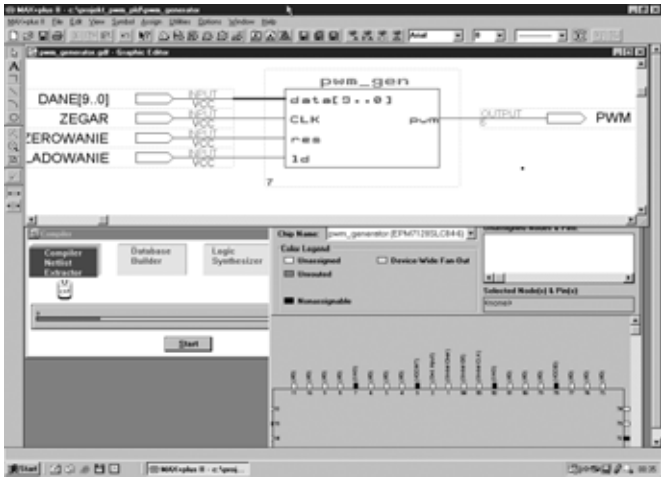
Wszystkie narzędzia programowe wykorzystane podczas przygotowywania artykułu są udostępniane przez producentów bezpłatnie (wymagana jest jedynie rejestracja i - w przypadku oprogramowania firmy Altera - bezpłatna aktualizacja co 6 miesięcy licencji).

System Max+Plus II oraz syntezer AAS są dostępne pod adresem: https://www.altera.com/support/software/download/altera_design/mp2_baseline/dnl-baseline.jsp,

System WebPack ISE jest dostępny pod adresem: http://www.xilinx.com/support/download.htm.



Rys. 7. Widok okna edytora schematów z pakietu WebPack ISE z symbolem generatora PWM



Rys. 8. Widok okna edytora schematów w Max+Plus II z symbolem generatora PWM

List. 1. Opis w języku VHDL 4-bitowego generatora PWM

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity pwm_gen is port (
  data           : in std_logic_vector(3 downto 0);
  clk, res, ld   : in std_logic;
  pwm            : out std_logic
                 );
end pwm_gen;

architecture Behavioral of pwm_gen is

signal data_int_cmp       : std_logic_vector(3 downto 0);
signal data_int, cnt_out  : std_logic_vector(3 downto 0);
signal data_int2          : std_logic_vector(3 downto 0);
signal res_pwm_o, q, co   : std_logic;
signal ld_int, pwm_cn     : std_logic;

begin
-- rejestr wejsciowy pierwszego stopnia
process (ld, res)
begin
  if res = '1' then
     data_int <= "0000";
  elsif ld = '1' then
     data_int <= data;
  end if;
end process;

-- rejestr wejsciowy drugiego stopnia
process (co, res)
begin
  if res = '1' then
     data_int_cmp <= "0000";
  elsif rising_edge(co) then
     data_int_cmp <= data_int;
  end if;
end process;

-- licznik
process (clk, res)
begin
  if res = '1' then
     cnt_out <= "0000";
  elsif rising_edge(clk) then
     cnt_out <= cnt_out + 1;
  end if;
end process;

-- generowanie sygnalu przeniesienia z licznika
process (clk, res)
begin
  if res = '1' or cnt_out < "1111"  then
     co <= '0';
  elsif rising_edge(clk) and cnt_out = "1111" then
     co <= '1';
  end if;
end process;

-- komparator
process (data_int_cmp, cnt_out)
begin
  if cnt_out = "0000" then
     res_pwm_o <= '0';
  elsif  data_int_cmp >= cnt_out then
     res_pwm_o <= '1';
  else
     res_pwm_o <= '0';
  end if;
end process;

-- przerzutnik PWM
process (clk, res, res_pwm_o)
begin
  if res = '1' then
     q <= '0';
  elsif rising_edge(clk) then
     q <= res_pwm_o;
  end if;
end process;

pwm <= q;

end Behavioral;
```

wyjść licznika. Wprowadzenie w opisie komparatora, wydawałoby się zbędnego, warunku `else res_pwm_o <= '0';` zapobiega zsyntezowaniu na wyjściu *res_pwm_o* przerzutnika podtrzymującego stan wyjścia.

Na list. 1 jest opis 4-bitowego generatora PWM. Łatwo oszacować, że oferowana przez niego rozdzielczość jest niezbyt wielka i z pewnością zbyt mała, aby można go było potraktować „poważnie". Wersja taka powstała wyłącznie dla wygody testowania - analiza działania automatu o 16 stanach jest przecież zdecydowanie łatwiejsza niż na przykład automatu 256-stanowego. Zwiększenie rozdzielczości nastaw sygnału wyjściowego PWM jest możliwe poprzez zwiększenie długości licznika, komparatora i rejestrów. Zmiana tych parametrów wymaga każdorazowo ingerencji w plik źródłowy projektu i zmiany przykładowej wartości „3" w deklaracji `std_logic_vector(3 downto 0)` na wybraną liczbę. Nie jest to rozwiązanie wygodne, ani eleganckie i często prowadzi do błędów uniemożliwiających kompilację projektu. W związku z tym powstała alternatywna wersja projektu z list. 1, różniąca się od pierwowzoru możliwością łatwej parametryzacji. Zamieszczono ją na **list. 2**.

Jak można zauważyć, w deklaracji jednostki projektowej zastosowano klauzulę `generic`, która umożliwia zdefiniowanie jej parametrów ogólnych. W prezentowanym przykładzie służy ona do określenia rozdzielczości generatora PWM, co uzyskano definiując stałą *pwm_res*, której jest przypisywana wartość `positive:= n` (w przykładzie n = 4). Parametr `n` może być liczbą całkowitą dodatnią, co ustalono poprzez określenie podtypu tej liczby jako *positive* (może przyjmować wartości od 1 do 2147483647). Przypisanie temu parametrowi żądanej wartości jest jedynym zabiegiem (poza rekompilacją projektu) niezbędnym podczas zmiany rozdzielczości generatora PWM. Zmiana pozostałych wartości jest wykonywana automatycznie przez program wykonujący syntezę.

Implementacja

Wykorzystanie języka opisu VHDL zapewnia możliwość względnie łatwego przenoszenia projektu pomiędzy systemami służącymi do syntezy logicznej i implementacji w strukturach PLD różnych producentów. Prezentowany generator zaimplementowano w dwóch układach: XC95108-15 w obudowie PLCC84 oraz EPM7128-12 (także w obudowie PLCC84). Jako narzędzia projektowe wykorzystano:

- udostępniony bezpłatnie przez firmę Xilinx pakiet WebPack ISE 5.2 z niezłym kompilatorem VHDL i zewnętrznym symulatorem ModelSim firmy Mentor Graphics,

WebPack ISE dla Windows XP

Wadą najnowszej wersji pakietu WebPack ISE (5.2) jest jego kompatybilność wyłącznie z Windows XP. Użytkownicy starszych wersji Windows muszą korzystać ze starszych wersji pakietu, które charakteryzują się niewiele gorszymi możliwościami.

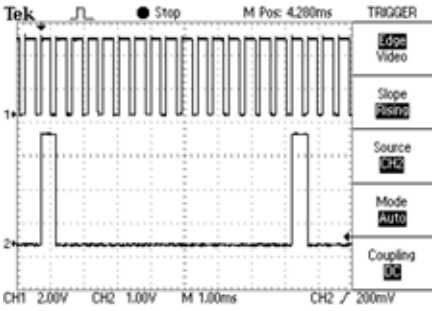
- zestaw bezpłatnych narzędzi firmy Altera: kompletne środowisko projektowe Max+Plus II Baseline 10.2 oraz syntezer VHDL - Advanced Altera Synthesis (który zastąpił udostępniany do niedawna przez Mentor Graphics syntezer Leonardo Spectrum).

Ponieważ projekt składa się z jednego pliku zawierającego opis w języku VHDL, konfiguracja i przygotowanie projektu do kompilacji, niezależnie od systemu projektowego, jest bardzo proste. WebPack ISE zawiera komplet narzędzi niezbędnych do syntezy, implementacji i kompilacji projektu opisanego w VHDL. W zależności od przyjętego sposobu projektowania, użytkownik może włączyć plik tekstowy VHDL do struktury własnego projektu lub - jak sądzę jest wielu zwolenników tego sposobu projektowania - stworzyć graficzny element biblioteczny (*Processing for Current Source>Design Entry Utilities>Create Schematic Symbol* w Web Pack ISE 5.2) i po prostu narysować schemat logiczny generatora z wykorzystaniem gotowego „bloczka“ (**rys. 7**).

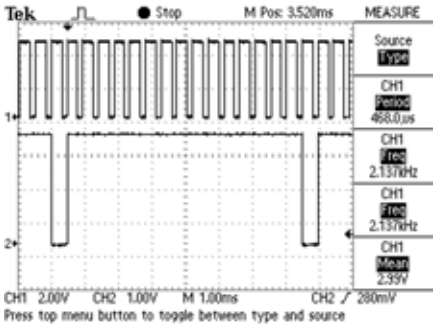
W nieco gorszej sytuacji znajdują się użytkownicy systemu Max+Plus II, który - co prawda - w wersji Student Edition udostępnia syntezer VHDL, ale jego możliwości są mocno ograniczone. Projektanci korzystający z układów firmy Altera mogą wykorzystać jako narzędzie projektowe nowszy system projektowy tej firmy Quartus II, ale jego popularność (głównie ze względu na trudniejszą obsługę i duże wymagania sprzętowe) jest znacznie mniejsza niż Max+Plus II. Skorzystanie z pakietu Max+Plus II jest jednak możliwe, ale w tym celu trzeba skorzystać z pomocy zewnętrznego syntezera VHDL firmy Altera (bezpłatny program Advanced Synthesis), za pomocą którego jest tworzona lista połączeń EDIF. Następnie, już za pomocą Max+Plus II, z pliku tekstowego w formacie EDIF można utworzyć na przykład symbol schematowy (**rys. 8**), który można następnie wykorzystać w kolejnych projektach w taki

List. 2. Opis parametryzowanego generatora PWM

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity pwm_gen is
generic (
     pwm_res     : positive:= 4  -- positive oznacza 1...2147483647
         );
port (
   data          : in std_logic_vector(pwm_res-1 downto 0);
   clk, res, ld  : in std_logic;
   pwm           : out std_logic
                       );
end pwm_gen;

architecture Behavioral of pwm_gen is

signal data_int_cmp       : std_logic_vector(pwm_res-1 downto 0);
signal data_int, cnt_out  : std_logic_vector(pwm_res-1 downto 0);
signal data_int2          : std_logic_vector(pwm_res-1 downto 0);
signal res_pwm_o, q, co   : std_logic;
signal ld_int, pwm_cn     : std_logic;

constant zero: std_logic_vector(pwm_res-1 downto 0):= (others => '0');
constant ff: std_logic_vector(pwm_res-1 downto 0):= (others => '1');

begin
-- rejestr wejsciowy pierwszego stopnia
process (ld, res)
begin
   if res = '1' then
      data_int <= (others => '0');
   elsif ld = '1' then
      data_int <= data;
   end if;
end process;

-- rejestr wejsciowy drugiego stopnia
process (co, res)
begin
   if res = '1' then
      data_int_cmp <= (others => '0');
   elsif rising_edge(co) then
      data_int_cmp <= data_int;
   end if;
end process;

-- licznik
process (clk, res)
begin
   if res = '1' then
      cnt_out <= (others => '0');  -- zapis alternatywny (cnt_out'range => '0');
   elsif rising_edge(clk) then
      cnt_out <= cnt_out + 1;
   end if;
end process;

-- generowanie sygnalu przeniesienia z licznika
process (clk, res)
begin
   if res = '1' or cnt_out < ff  then
      co <= '0';
   elsif rising_edge(clk) and cnt_out = ff then
      co <= '1';
   end if;
end process;

-- komparator
process (data_int_cmp, cnt_out)
begin
   if cnt_out = zero then
      res_pwm_o <= '0';
   elsif  data_int_cmp >= cnt_out then
      res_pwm_o <= '1';
   else
      res_pwm_o <= '0';
   end if;
end process;

-- przerzutnik PWM
process (clk, res, res_pwm_o)
begin
   if res = '1' then
      q <= '0';
   elsif rising_edge(clk) then
      q <= res_pwm_o;
   end if;
end process;

pwm <= q;

end Behavioral;
```



Rys. 9. Przebiegi: zegarowy i wyjściowy dla zadanego współczynnika wypełnienia 1/16 (4-bitowy generator PWM)



Rys. 10. Przebiegi: zegarowy i wyjściowy dla zadanego współczynnika wypełnienia 15/16 (4-bitowy generator PWM)

sam sposób jak pozostałe (także te dostarczone wraz z systemem projektowym) elementy biblioteczne.

Po implementacji prezentowanego projektu i przy założeniu, że generator PWM ma rozdzielczość 10-bitową (czyli dla *pwm_res* = 10), okazało się, że w układzie XC95108 (zawiera 108 makrokomórek) wykorzystano 35 makrokomórek, w tym 22 rejestry, a maksymalna częstotliwość taktowania generatora (dla optymalizacji „powierzchniowej" i układu z sufiksem -15) wynosi 71,429 MHz (wynik nie był weryfikowany w praktyce). Podobne wyniki uzyskano w przypadku implementacji projektu w układzie EPM7128. Wykorzystano także 35 makrokomórek (spośród 128 dostępnych), a maksymalna częstotliwość taktowania w przypadku układu oznaczonego sufiksem -12 wynosiła 73,6 MHz (wynik symulacji, nie weryfikowany w praktyce).

Uzyskane efekty

Podczas opracowywania projektu, wszelkie testy sprzętowe były prowadzone na uniwersalnym zestawie ewaluacyjnym ZL1PLD (udostępniony przez firmę BTC), który dzięki specjalnej konstrukcji umożliwia stosowanie dowolnych układów PLD. Standardowo, są w nim stosowane układy CPLD zasilane napięciem 5 V. Polityka cenowa producentów układów powoduje, że najtańsze są układy zasilane napięciem 3,3 V z portami I/O przystosowanymi do współpracy z układami zasilanymi napięciem 5 V. I tak przykładowo, koszt implementacji prezentowanego generatora PWM wynosi:

- w układzie XC95108-15 w obudowie PLCC84 (zasilanie 5 V): 16,47 zł brutto,
- w układzie XC95144XL-15 w obudowie TQFP100 (zasilanie 3,3 V): 8,65 zł brutto,
- w układzie EPM7128-12 w obudowie PLCC84 (zasilanie 5 V): 19 zł brutto,
- w układzie EPM3128A-10 w obudowie TQFP100 (zasilanie 3,3 V): 14,51 zł brutto.

Zestawienie to przygotowano na podstawie cen dystrybucyjnych brutto w Polsce (zakup pojedynczych sztuk). Do zestawienia dobrano układy najbardziej do siebie zbliżone pod względem liczby wbudowanych makrokomórek, ponieważ nie są dostępne ścisłe odpowiedniki w wersjach zasilanych napięciami 3,3 V oraz 5 V.

Na **rys. 9** i **10** pokazano przykładowe wyniki działania 4-bitowego generatora PWM zaimplementowanego w układzie PLD.

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

Kłopoty wynikające z niezgodności wersji

Udostępniony na naszej stronie internetowej projekt dla systemu WebPack ISE 5.2 nie jest kompatybilny ze starszymi wersjami systemu projektowego. Wynika to z niezgodności formatu plików **.npl* i zastosowanej przez twórców systemu projektowego struktury plików pomocniczych.