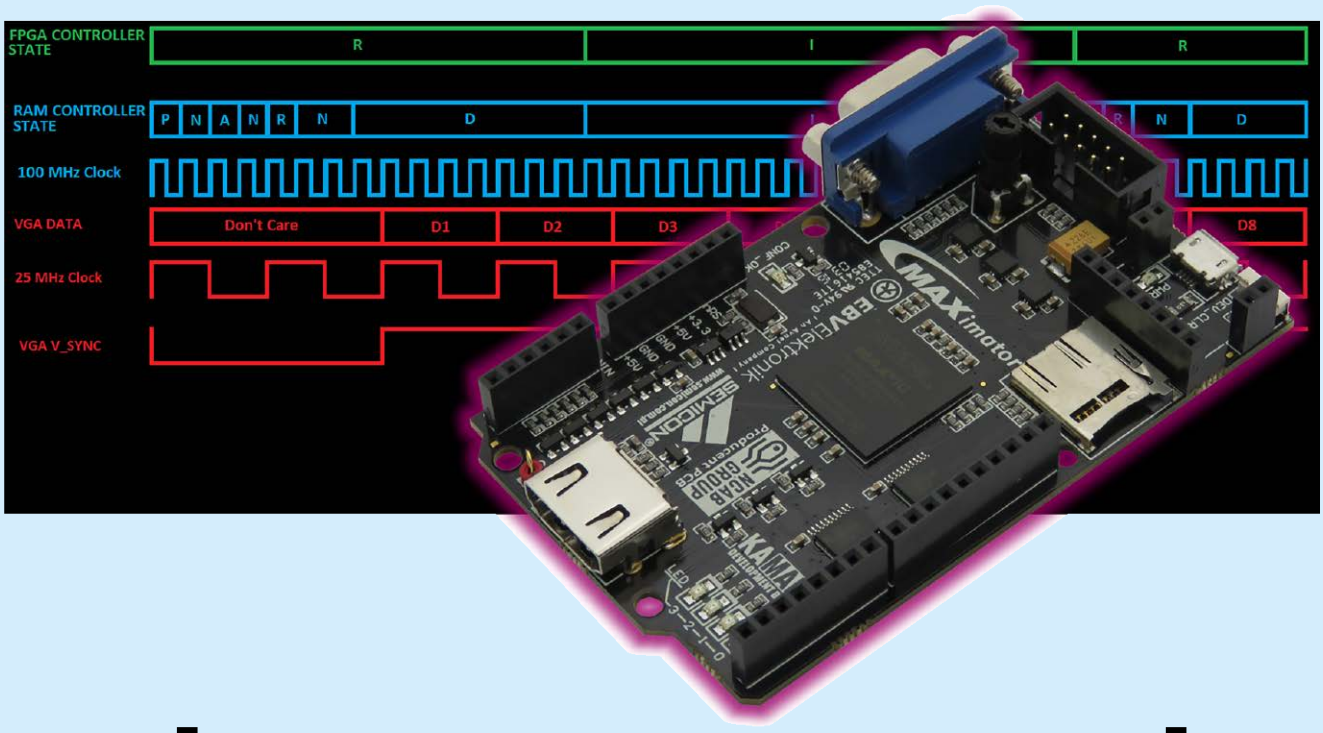


Podstawy generowania grafik w FPGA za pomocą VHDL (3)

Generator obrazu VGA – przygotowanie generatora taktującego

Możemy teraz przystąpić do pisania kodu sprzętowego generatora obrazu VGA. Zanim do tego przejdziemy musimy ustalić, z jaką rozdzielczością obrazu chcemy pracować oraz z jaką częstotliwością odświeżania. Otóż my przyjmiemy następujące parametry: rozdzielczość 640×480 pikseli i częstotliwość odświeżania 60 Hz. Po przyjęciu tych założeń przejdźmy do ustalenia częstotliwości zegara graficznego, który będzie napędzał nasz generator.

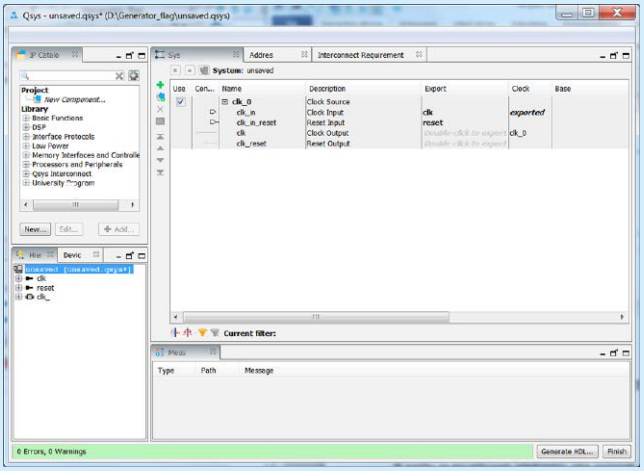
Częstotliwości generatora są ściśle zależne od wartości wspomnianych przed chwilą parametrów. Dla rozdzielczości 640x480 punktów i przy częstotliwości odświeżania 60 Hz częstotliwość linii wynosi 31,46875 kHz, a częstotliwość piksela 25,175 MHz. Jak się okazuje, w praktyce stosuje się równe 25 MHz, a co najważniejsze takie „zaokrąglenie" nie wpływa na działanie sterownika (niektóre monitory co najwyżej pokażą, że będziemy mieli do czynienia z częstotliwością odświeżania 59 Hz zamiast 60 Hz). Niemniej, wracając dochodzimy do pierwszego problemu, jaki musimy rozwiązać. Otóż MAXimator dysponuje wyłącznie zegarem o częstotliwości 10 MHz, podczas gdy my potrzebujemy zegara o częstotliwości 25 MHz. Na szczęście nic straconego – pomoże nam w tym względzie pętla PLL, która umożliwi nam wygenerowanie przebiegu o takiej częstotliwość, jaką chcemy. Tym samym musimy ją dodać do projektu. W tym celu wchodzimy w menu *Tools* → *Qsys*. W wyniku tego zostanie wyświetlone okno kreatora QSYS, jak pokazano na **rysunku 40**.

Aby dodać pętlę PLL, w pierwszej kolejności zaznacz zegar, który jest obecny w głównym oknie (tzn. najedź na niego i kliknij lewym przyciskiem myszy) – **rysunek 41**. Następnie kliknij w ikonę z czerwonym „iksem" po lewej stronie, w wyniku czego usuniesz domyślnie wstawiony zegar. Po tej operacji okno powinno wyglądać w sposób pokazany na **rysunku 42**. W dalszej kolejności w okienku wyszukiwania, które znajduje się na górze zakładki IP Catalog wpisz ALTPLL, a po wyświetleniu listy możliwych pętli w oknie poniżej do wyboru zaznacz wybór „Avalon ALTPLL". Po zaznaczeniu wyboru kliknij „Add" – **rysunek 43**. Dodatkowo zostanie wyświetlone okno pokazane na **rysunku 44**.

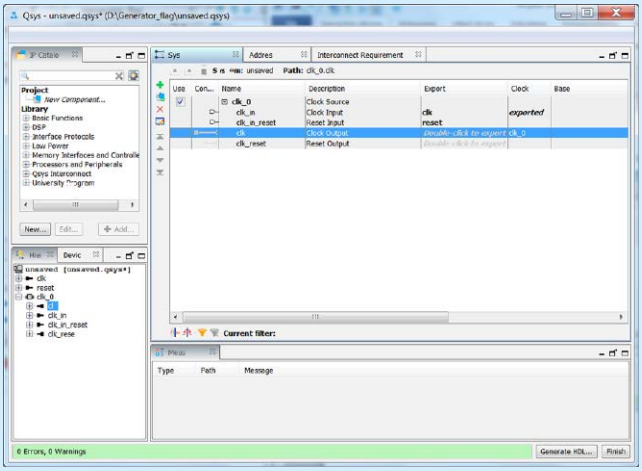
Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: **ftp://ep.com.pl**, user: **33372**, pass: **6nwd5fit4**

Jak wspomnieliśmy wcześniej maXimator dysponuje zegarem o częstotliwości 10 MHz, a zatem musimy taką wartość wpisać. Tym samym w oknie „What is the frequency of the inclk0 input" zamiast domyślnych 100.000 MHz wpisujemy wartość 10 MHz. Tym samym okno po zmianie wartości wygląda jak na **rysunku 45**. Pozostałe ustawienia zostawiamy bez zmian i klikamy „Next >". Zostanie wyświetlone okno z **rysunku 46**, w którym to możemy zdecydować, jakie dodatkowe wejścia i wyjścia będziemy używać. Jak widać, mamy tu do czynienia z wejściem asynchronicznego zerowania pętli oraz z wyjściem „locked", które wskazuje czy wyjście pętli jest czynne czy też nie. Te obydwie rzeczy nam się nie przydadzą, dlatego też odznacz ustawienia, jakie zaznaczone są w tym oknie.

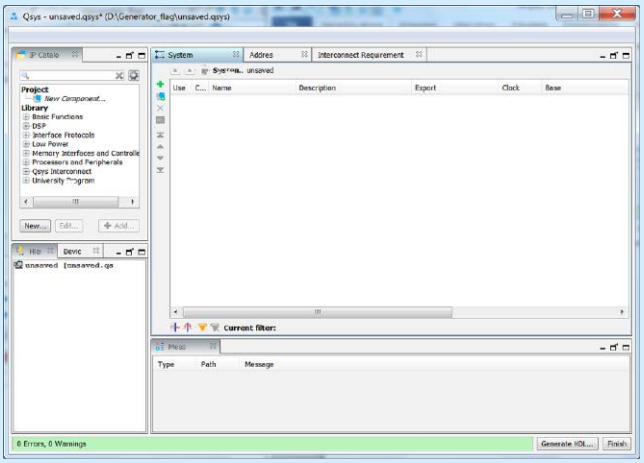
Po odznaczeniu wspomnianych ustawień kliknij „Next >". Zostanie pokazane kolejne okno, którego widok znajduje się na **rysunku 47**. W oknie tym nic nie zmieniaj (pozostaw ustawienia takie jakie tu są), a tym samym kliknij „Next >". Pojawi się w wyniku okno, które też należy pominąć, kliknij ponownie „Next >". Następne okno, jakie zostanie wyświetlone również pomiń (nie zmieniaj tu ustawień). W ten sposób dochodzimy do najważniejszego



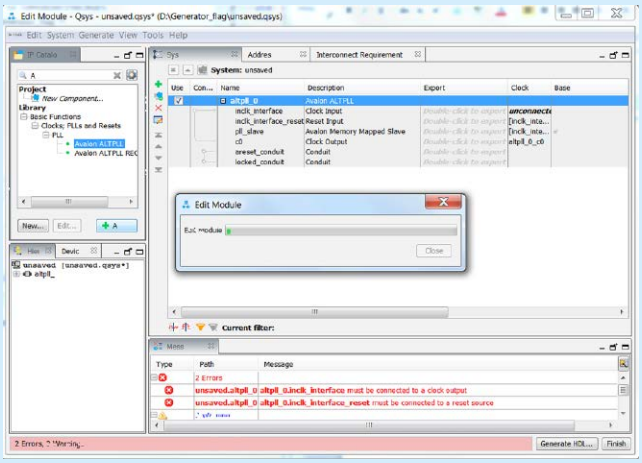
Rysunek 40. Okno kreatora Qsys



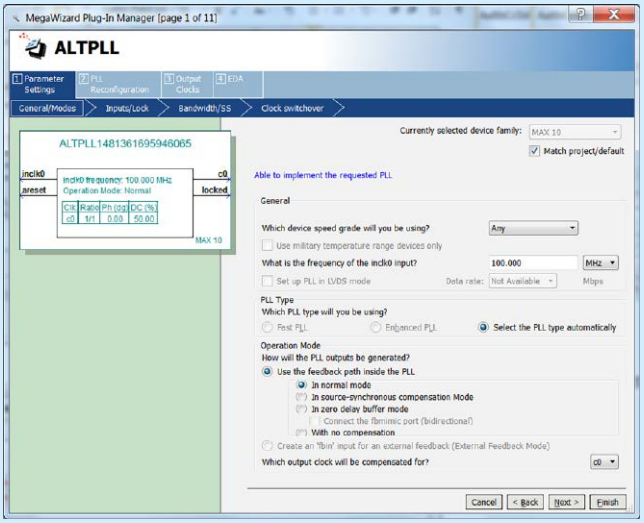
Rysunek 41. Dodawanie pętli PLL



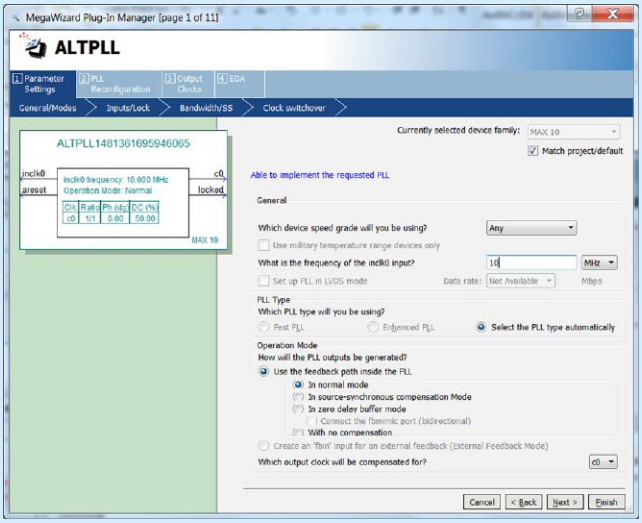
Rysunek 42. Okno kreatora po modyfikacji zegara



Rysunek 43. Dodanie modułu pętli PLL



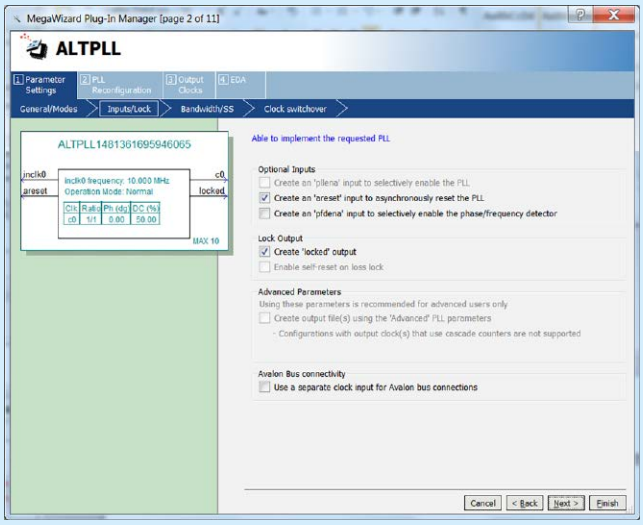
Rysunek 44. Okno parametrów modułu pętli PLL



Rysunek 45. Okno parametrów PLL po zmianie częstotliwości

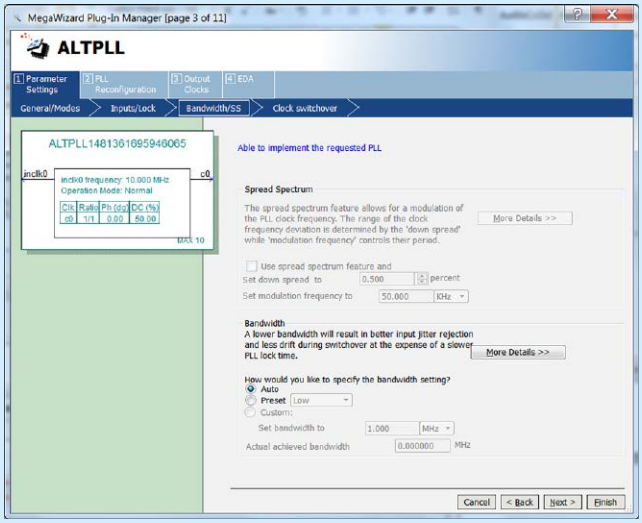
z okien – **rysunek 48**. W oknie tym będziemy określali docelową częstotliwość, która ma być uzyskana z zegara 10 MHz (czyli nasze wcześniej określone 25 MHz). Aby to zrobić zaznacz „Enter output clock frequency”, po czym w oknie „Requested Settings” wpisz „25”, tak jak to przedstawione zostało na **rysunku 49**.

W opisany wyżej sposób określiliśmy, że konfigurowana przez nas pętla PLL będzie zamieniała zegar 10 MHz w zegar 25 MHz.

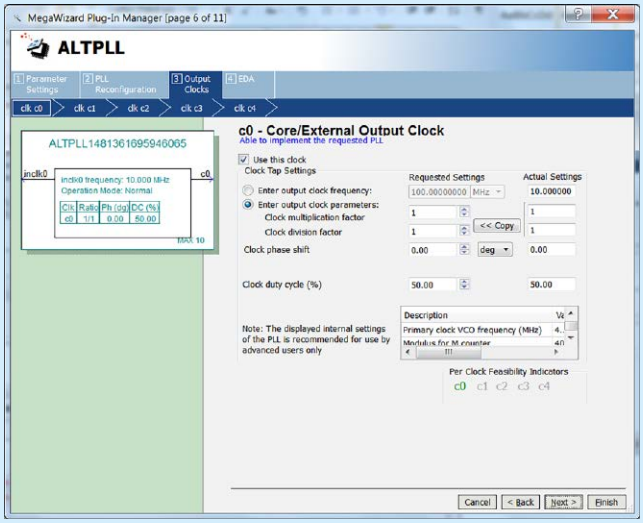


Rysunek 46. Wybranie dodatkowych wejść/wyjść

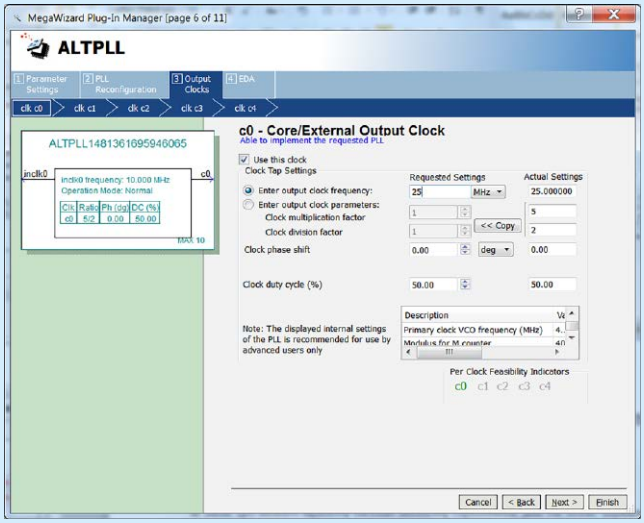
Po wprowadzeniu wspomnianej wartości kliknij „Next >” – w wyniku zostanie pokazane okno jak na **rysunku 50**. Okno to to nic innego jak opcje związane z następnym wyjściem, jakie możemy wstawić do pętli (ta pętla ma możliwość generowania na podstawie jednego przebiegu zegarowego generowania kilku innych). Jednak w naszym przypadku potrzebny jest tylko jeden zegar,



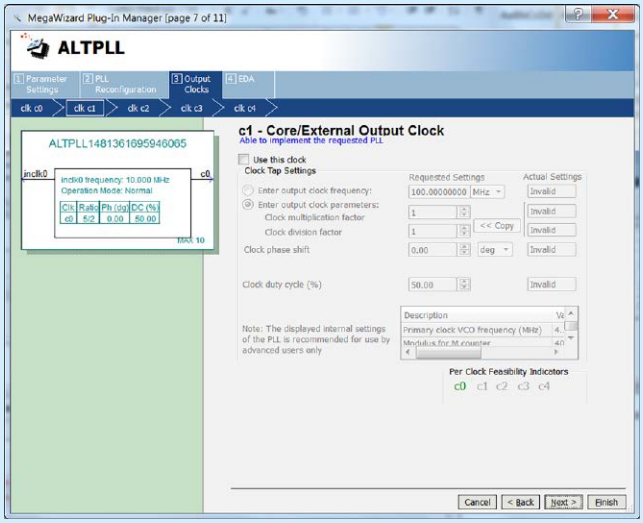
Rysunek 47. Okno dające możliwość wyboru modulacji częstotliwości pętli



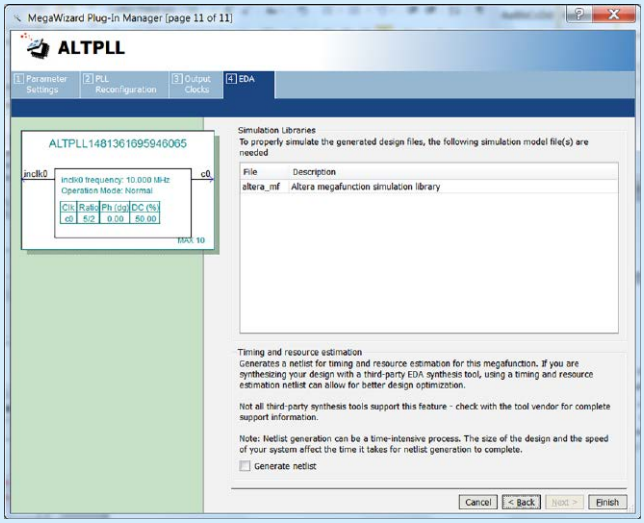
Rysunek 48. Okno ustawień częstotliwości docelowej



Rysunek 49. Zmiana częstotliwości generatora



Rysunek 50. Okno opcji kolejnego wyjścia pętli

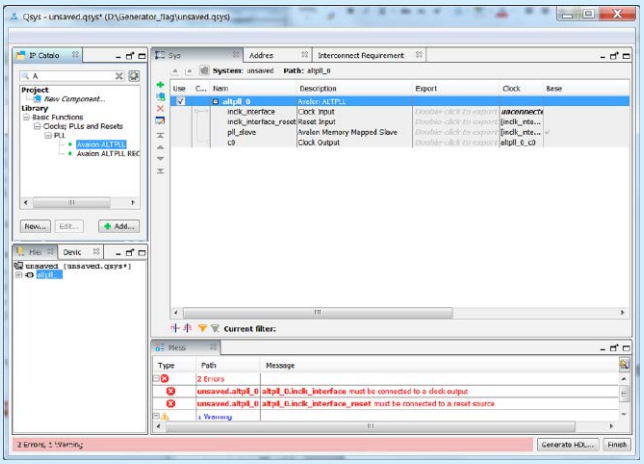


Rysunek 51. Okno wyboru pliku z symulacjami

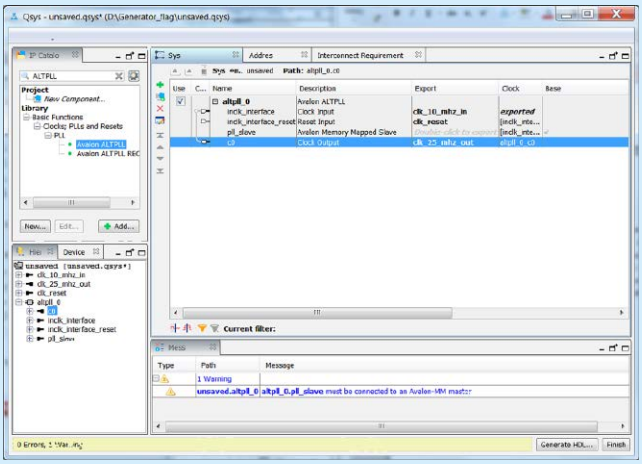
dlatego nic tu nie zmieniaj i kliknij „Next >”, co spowoduje pojawienie się kolejnego identycznego okna, w którym to możemy określić, aby konfigurowana pętla miała kolejne wyjście. Jednak i to nas nie interesuje, dlatego następne okno również pomiń. I tak z pozostałymi dwoma identycznymi oknami (tu też nic nie zmieniaj i kliknij „Next >”. Kiedy przeklikasz wspomniane okna powinieneś dojść do okna pokazanego na **rysunku 51**. W oknie tym mamy określenie tego do jakiego pliku zostaną wyeksportowane biblioteki symulacyjne pętli oraz jest dostępna opcja, aby wygenerować opcjonalnie netlistę dla tej pętli. Jednak nam to nie będzie potrzebne, dlatego nic tu nie zmieniaj i kliknij „Finish”. W ten sposób zakończyliśmy wstępną konfigurację pętli, a główne okno zaś po jej dokonaniu wygląda jak na **rysunku 52**.

Widzimy, że po wstawieniu pętli mamy jeszcze dwa błędy do rozwiązania. Są one spowodowane tym, że nie „wyprowadziliśmy” jeszcze wejść i wyjść pętli. Zatem uczyńmy to. Aby to zrobić, w kolumnie „Export” kliknij dwukrotnie lewym przyciskiem myszy w pozycję „Double-click to export”, która znajduje się przy pozycji „Clock Input” i wprowadź nazwę „clk_10_mhz_in”. Analogicznie zrób w wypadku pozycji „Clock Output” wprowadzając nazwę „clk_25_mhz_out” i „Reset Output” wprowadzając nazwę „clk_reset”. W wyniku tego okno powinno wyglądać w sposób pokazany na **rysunku 53**.

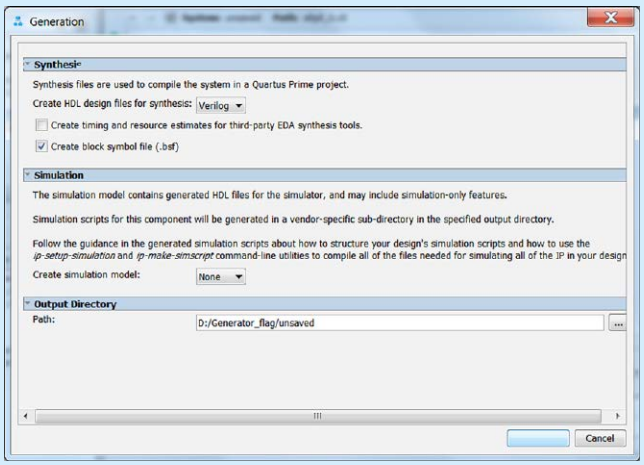
Uwaga! Ważne jest to, w jakiej kolejności zmieniamy nazwy (zaleca się, aby zrobić to tak jak zostało przed chwilą opisane – inaczej w dalszej części przy wstawianiu pętli do kodu pojawią się



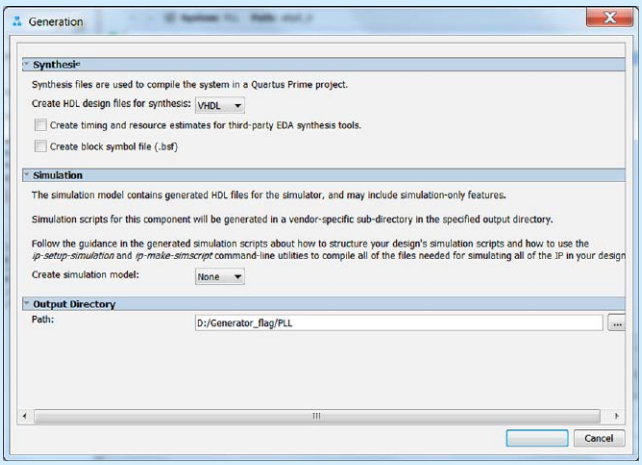
Rysunek 52. Okno kreatora PLL po zakończeniu konfiguracji



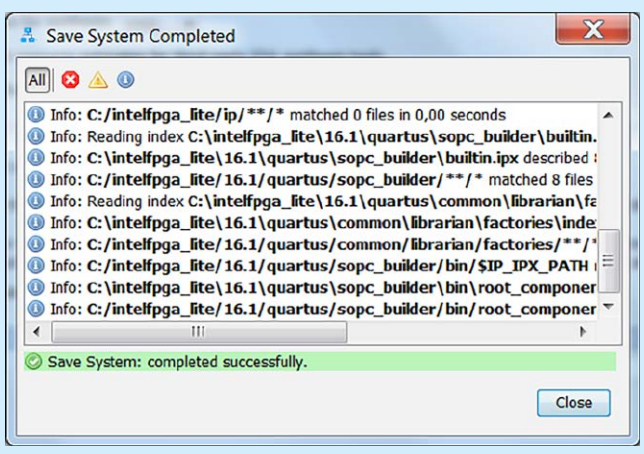
Rysunek 53. Konfigurowanie wejść/wyjść pętli



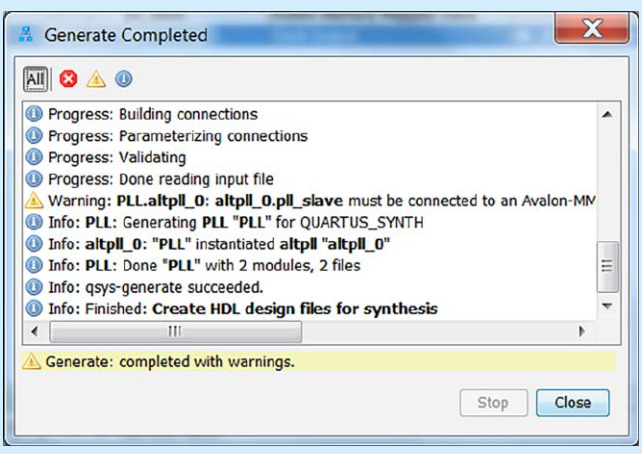
Rysunek 54. Generowanie pliku pętli PLL



Rysunek 55. Okno konfiguracji pętli po zmianach

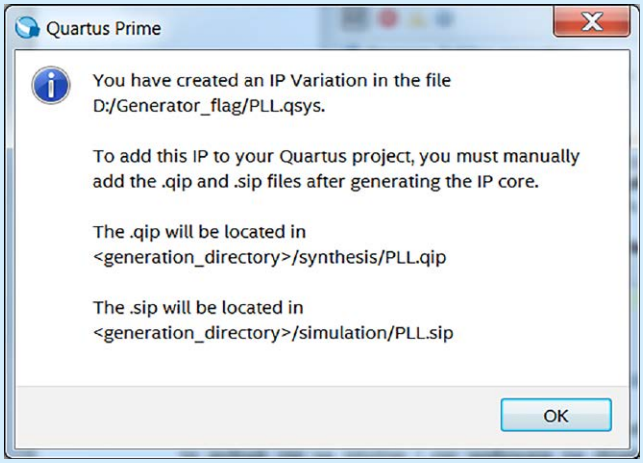


Rysunek 56. Komunikat o zapamiętaniu pliku



Rysunek 57. Komunikat o pomyślnym wygenerowaniu pętli

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 33372, pass: 6nwd5fk4



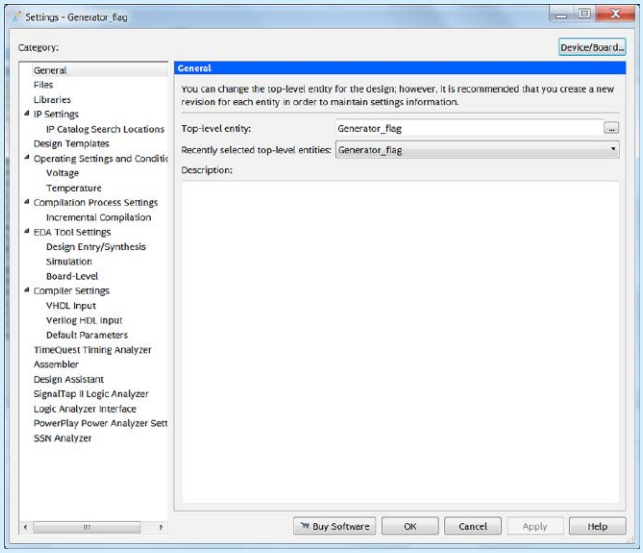
Rysunek 58. Informacja o zapisanych plikach pętli

problemy!). Niemniej po tej czynności skonfigurowaliśmy w kompletny sposób pętlę PLL. Pozostaje nam jeszcze ją wygenerować w postaci plików i kodu, które następnie dołączymy do projektu. W tym celu kliknij „Generate HDL”, w wyniku czego zostanie wyświetlone okno pokazane **rysunku 54**.

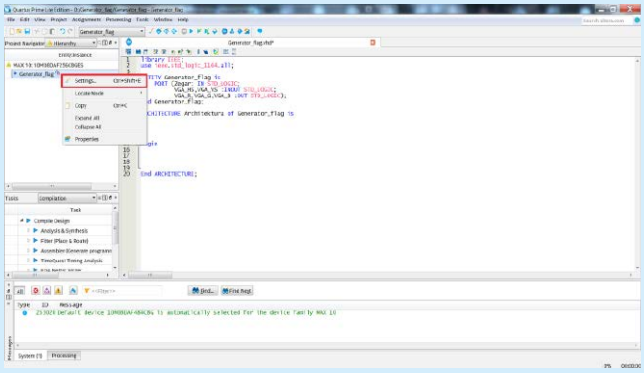
Jak widać możemy wybrać jakim językiem ma być opisana pętla PLL - VHDL lub Verilog. Wybór języka jest w zasadzie bez znaczenia, niemniej dla celów dydaktycznych weźmiemy pierwszy z nich. Wobec tego zmień wybór „Verilog” znajdujący się przy pozycji „Create HDL design files for synthesis” na „VHDL”. Ponadto domyślnie zaznaczona jest opcja generacji symbolu pętli dla schematów blokowych. Nam jednak to nie będzie potrzebne, dlatego odznacz opcję „Create block symbol file (.bsf). Na koniec w oknie znajdującym się przy pozycji Path zmień „unsaved” na „PLL”. W wyniku tych zmian okno będzie wyglądać jak na **rysunku 55**.

Tym samym po wprowadzeniu wspomnianych wyżej zmian kliknij „Generate”, w wyniku czego program zapyta o to czy zapisać zmiany. Kliknij „Save”, aby zapisać zmiany, wpisując w oknie zapisu „PLL.qsys”. Po zmianie nazwy kliknij „Save”. W ten sposób w pierwszej kolejności zostaną zapisane ustawienia związane z pętlą, co potrwa chwilę, a co będzie nam przedstawiało odrębne okno, w którym to po zakończeniu zapisu pojawi się komunikat pokazany na **rysunku 56**.

Tym samym po zakończeniu zapisu okno to należy zamknąć. W wyniku tego zostanie wyświetlone kolejne okno związane z generowaniem plików dotyczących naszej pętli PLL, a tym samym rozpocznie się ten proces. Oczywiście potrwa to chwilę, po czym



Rysunek 60. Okno ustawień opcji generatora



Rysunek 59. Wybór menu nastaw

po zakończeniu we wspomnianym odrębnym oknie pojawi się komunikat pokazany na **rysunku 57**.

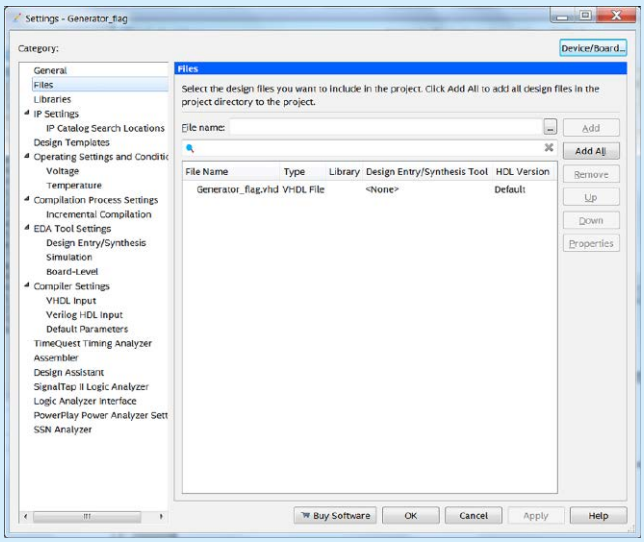
Komunikat ten mówi nam, że generowanie pętli przebiegła pomyślnie, choć z pewnymi uwagami, które nie są dla istotne i nie wpływają na działanie pętli, dlatego nie należy się nimi przejmować. Wobec tego kliknij „Close”, aby zamknąć to okno. Tym samym mamy wszystko, co nam potrzebne odnośnie do pętli. Zatem w głównym oknie, jakie nam pozostało kliknij „Finish”, w wyniku czego pojawi się nam komunikat informujący o tym, że wygenerowaliśmy pliki pętli PLL oraz gdzie się one znajdują (**rysunek 58**).

Pozostaje nam jeszcze wstawienie pętli do projektu tzn. wstawienie pliku konfiguracyjnego pętli o rozszerzeniu .qip oraz stosownego kodu, w którego wyniku w projekcie pętla ta zostanie użyta. Zatem w pierwszej kolejności kliknij prawym przyciskiem myszy w nazwę projektu tj. „Generator_flag” w oknie „Entity:Instance”, po czym z menu, które się pojawi wybierz „Settings” (**rysunek 59**). W wyniku tego zostanie pokazane okno jak na **rysunku 60**. W oknie tym przejdź do zakładki „Files”, w wyniku czego okno to będzie wyglądać jak na **rysunku 61**.

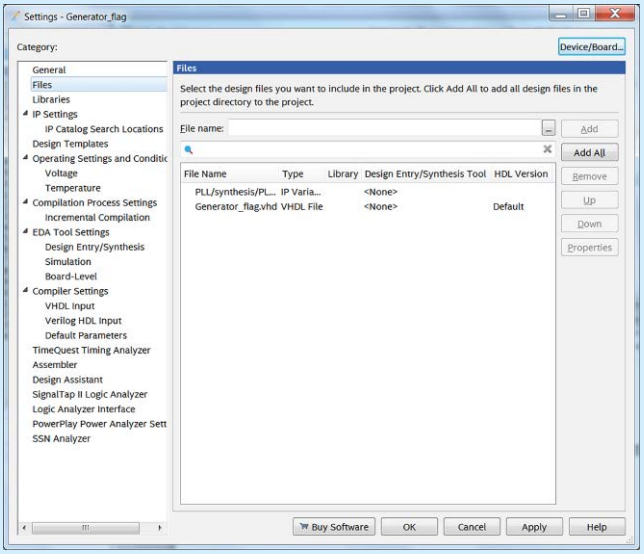
Po przejściu do wskazanej zakładki kliknij w przycisku z trzema kropkami („…”) i odszukaj plik „PLL.qip” (w naszym przypadku plik ten zawarty jest w lokalizacji D:\Generator_flag\PLL\synthesis). Po jego odnalezieniu należy go zaznaczyć i kliknąć „Otwórz”. W wyniku tego pętla została dołączona do projektu, a okno ustawień przybiera wtedy formę pokazaną na **rysunku 62**.

Po dodaniu pliku kliknij „Apply”, po czym zamknij te okno, klikając „OK”. Na koniec w pliku, który masz otwarty w programie (tj. Generator_flag.vhd) między wierszami „ARCHITECTURE Architektura of Generator_flag is” a „Begin” dodaj następujący kod

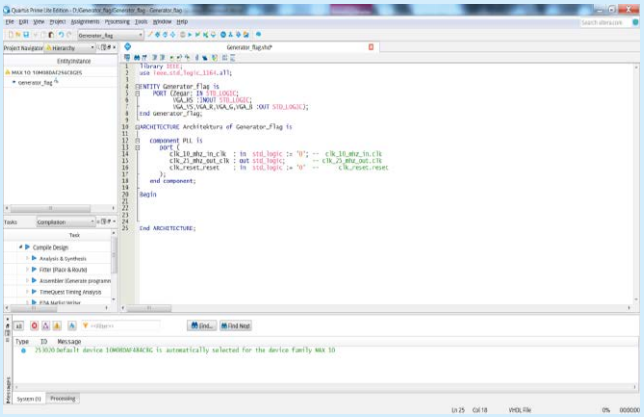
```
component PLL is
```



Rysunek 61. Wyświetlenie zakładki Files



Rysunek 62. Okno ustawień po dołączeniu pętli do projektu



Rysunek 63. Okno programu po dodaniu komponentu

```
  port (
    clk_10_mhz_in_clk  : in  std_logic :=
,0'; --  clk_10_mhz_in.clk
    clk_25_mhz_out_clk : out std_logic;
-- clk_25_mhz_out.clk
    clk_reset_reset    : in  std_logic := ,0'
--      clk_reset.reset
    );
end component;
```

Kod ten powstał na podstawie pliku PLL.vhd, który znajduje się w tym samym folderze co plik PLL.qip, a w którym to zawarty jest kod opisujący działanie pętli, w tym deklaracje wejść i wyjść (różnica jest tylko taka, że zamiast słowa „Entity” zostało użyte słowo „Component”). Tym samym nie bez przypadku wybraliśmy wcześniej, aby pętla została opisana w języku VHDL (w ten sposób bardzo łatwo bowiem uzyskaliśmy deklarację komponentu, bazując na wygenerowanym pliku). Po jego wprowadzeniu okno programu powinno wyglądać jak na **rysunku 63**. W ten sposób zadeklarowaliśmy pętlę w projekcie, ale jeszcze jej nie użyliśmy. Wobec tego między wierszami „Begin” a „End ARCHTECTURE” dodaj następujący kod:

```
C1: PLL port map (Zegar,Zegar_25_MHz,'0');
```

Dzięki temu pętla jest już w użyciu w projekcie. Powyższy kod to nic innego jak przyporządkowanie wszystkim wejściom i wyjściom pętli stosownych wejść i wyjść jakimi odznacza się nasz generator flag, co sprawia, że pętla jest niejako zawarta w generatorze. Tym samym widzimy, że na wejście pętli podajemy port wejściowy Zegar, którym to zegar 10 MHz znajdujący się w maXimatorze „wchodzi” do generatora. Jednak widzimy, że z kolei wejście resetu oraz wyjście pętli odznaczają się innymi „portami”, które nie zostały wcześniej określone w generatorze, tzn. na wyjściu pętli mamy wyprowadzony Zegar_25_MHz, natomiast na wejściu resetu mamy wpisaną stałą wartość 0. Takie podejście nie jest niczym nadzwyczajnym, bowiem poprzez stałą wartość 0 ustalamy, że pętla nie jest zresetowana (a więc pracuje normalnie), natomiast na wyjście pętli podaliśmy tzw. sygnał wewnętrzny, który będzie wykorzystywany w kodzie, a który możemy wyobrazić sobie jako „chwytak”, którym będziemy posługiwać się w celu opisania tego w jaki sposób będzie pracował nasz generator. Jednak musimy taki „chwytak” zadeklarować w kodzie. W tym celu między wierszami „ARCHITECTURE Architektura of Generator_flag is” a „Begin, przed kodem związanym z deklaracją pętli, to jest:

```
component PLL is
  port (
    clk_10_mhz_in_clk  : in  std_logic :=
'0'; --  clk_10_mhz_in.clk
    clk_25_mhz_out_clk : out std_logic;
-- clk_25_mhz_out.clk
    clk_reset_reset    : in  std_logic := '0'
--      clk_reset.reset
  );
end component;
```

zostawiając jedną pustą linijkę wprowadź następujące polecenia „signal Zegar_25_MHz :std_logic;”. Po tak dokonanym wprowadzeniu kodu zapisz plik Generator_flag.vhd poprzez kliknięcie w ikonę dyskietki. Tym samym w projekcie jest obecna i w dodatku pełnoprawnie używana pętla PLL. W ten sposób mamy rozwiązany problem związany z dostarczeniem do generatora zegara o określonej wcześniej częstotliwości.

Jakub Tyburski, WAT

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 3372, pass: 6nwd5fk4