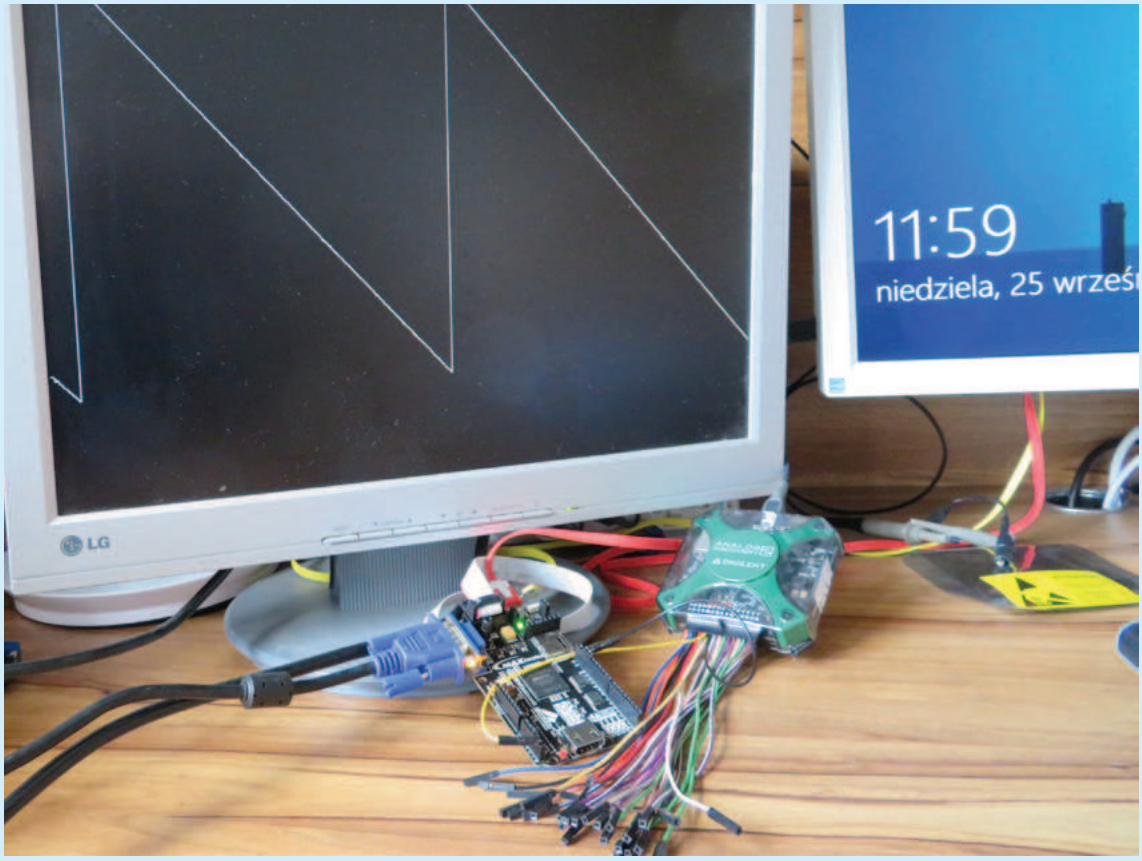


Pierwsze kroki z FPGA (5)

MAXimator Scope: oscyloskop z wyświetlaczem VGA

Układ FPGA na płytce MAXimator ma wbudowany przetwornik ADC z maksymalną prędkością próbkowania 1MSPS, oraz połączenia umożliwiające wyświetlanie obrazu na monitorze z interfejsem VGA. Daje to możliwość wykonania, bez konieczności posiadania dodatkowych komponentów, prostego oscyloskopu. W artykule tym przedstawione zostanie wykonanie prostego modelowego oscyloskopu z automatycznym i ręcznym wyzwalaniem akwizycji.

Na początku przygotujmy:

1. Płytkę MAXimator.
2. Programator USB Blaster (wchodzi w skład zestawu MAXimator).
3. Monitor z interfejsem VGA oraz stosowny kabel.
4. Generator sygnałów (w najprostszym wypadku wystarczy potencjometr, trochę bardziej zaawansowane rozwiązanie można wykonać w oparciu o dowolny mikrokontroler z wbudowanym przetwornikiem DAC, bądź połączonym z drabinką R-2R).
5. Komputer z zainstalowanym środowiskiem Quartus Prime (w czasie pisania artykułu użyto komputera z systemem Windows 7 oraz Quartus Prime Lite Edition 15.1.1).

Powinniśmy też mieć wiedzę na temat korzystania z płytki i oprogramowania, w szczególności dotycząca przeprowadzania syntezy programu, dodawania plików do projektu oraz wgrywania konfiguracji do układu za pomocą programatora.

Mając przygotowane wszystkie elementy możemy przystąpić do wykonania naszego prostego oscyloskopu. Oczywiście w podstawowej wersji daleko mu będzie do profesjonalnych rozwiązań, jednak przy odrobinie chęci i pracy można projekt ten znacząco rozwinąć i dodać dodatkowe funkcje, ale o tym na końcu.

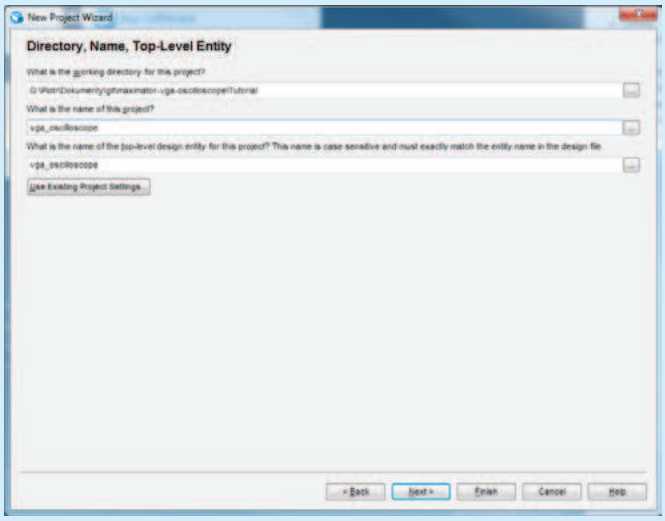
Architektura

Nasz oscyloskop możemy podzielić na kilka współpracujących ze sobą funkcjonalnych modułów. Oczywiście realizacji takiego oscyloskopu można wyobrazić sobie wiele, mniej lub bardziej skomplikowanych, jednak ja zdecydowałem się na przedstawienie prostego układu,

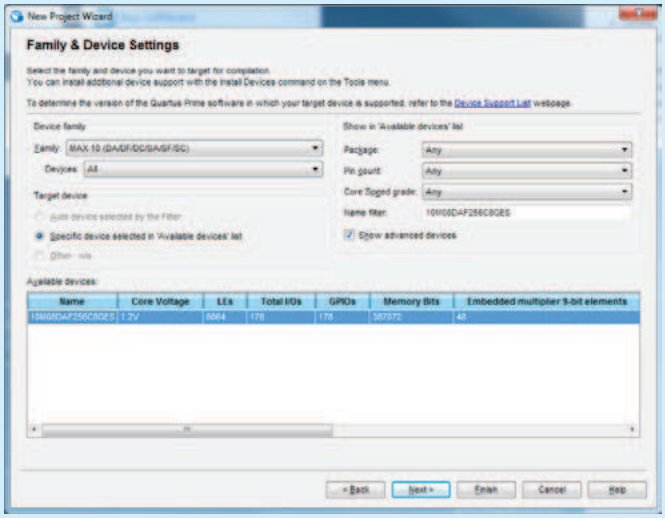
Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 22086, pass: 218655ee

z wyzwalaniem za pomocą przycisku na płytce i jedną pamięcią RAM, będącą pamięcią wyświetlania. Zanim przystąpimy do realizacji projektu omówmy działanie poszczególnych jego bloków.

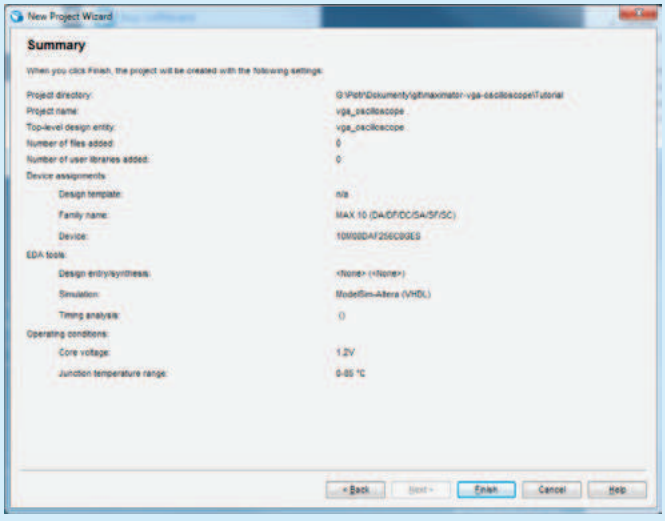
- **Pętla PLL** generuje przebiegi zegarowe o odpowiednich częstotliwościach, wymaganych do taktowania poszczególnych bloków systemu. Będą to częstotliwości: 10 MHz (powtórzenie częstotliwości wejściowej – zgodnie z dokumentacją układu przetwornik ADC musi być taktowany z wyjścia c0 pierwszej, dostępnej w układzie pętli PLL – u nas mamy dostępną tylko jedną pętlę PLL), 50 MHz (do taktowania modułu wyzwalana i zapisu do pamięci wyświetlania) oraz 65 MHz (częstotliwość, z jaką nadawać należy kolejne piksele dla otrzymania obrazu o rozmiarach 1024×768 pikseli odświeżanego z częstotliwością 60 Hz).
- **ADC** Blok przetwornika analogowo-cyfrowego, który będzie dokonywał pomiaru napięcia na wybranym wejściu.
- **ADC Command** Blok sterujący przetwornikiem. Jako iż wybierzemy później wariant bloku przetwornika bez układów sterujących, musimy nadawać do przetwornika ręcznie polecenie przeprowadzenia pomiaru oraz zatrzaskiwać dane – wynik pomiaru – kiedy te będą gotowe.
- **Trigger** Moduł wyzwalania. W naszym wypadku będzie on kontrolował zapis próbek do pamięci wyświetlania po przyciśnięciu przycisku. Umożliwi on nam także wybór tylko niektórych próbek pobranych przez przetwornik, aby zmniejszyć częstotliwość próbkowania (tzw. decymacja).
- **Display RAM** 2-portowa pamięć RAM, będąca w tym modelu pamięcią danych, które mają zostać wyświetlone na ekranie. Port zapisu będzie zapisywał próbki według sygnałów sterujących z modułu wyzwalania, natomiast port odczytu posłuży do odczytania odpowiednich danych, kiedy będzie konieczność wyświetlenia odpowiedniego elementu obrazu.
- **Line Drawer** Moduł rysowania linii. Jak sama nazwa wskazuje, posłuży on nam do narysowania linii (przebiegu) na ekranie. W zasadzie, jak później zobaczymy, będzie on miał za zadanie jedynie łączyć linią pionową kolejne punkty (wartości próbek).
- **VGA Driver** Moduł sterowania wyświetlaniem VGA. Moduł ten wygeneruje odpowiednie sygnały sterujące i będzie zliczał piksele obrazu, wskazując w ten sposób dane punktu obrazu, który ma mu zaprezentować moduł rysowania linii.



Rysunek 1. Kreator projektu – wybór lokalizacji i wpisanie nazw



Rysunek 2. Kreator projektu – wybór układu



Rysunek 3. Kreator projektu – wyświetlenie podsumowania

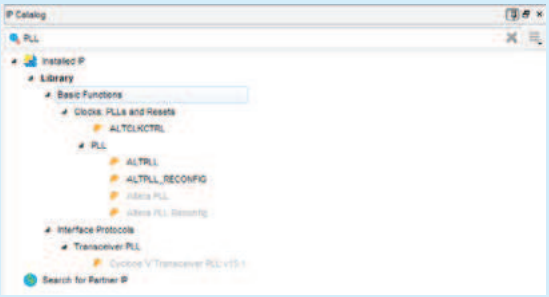
Utworzenie projektu

Pracę rozpoczynamy od utworzenia projektu, klikając na *File –> New Project Wizard*. W pierwszym wyświetlonym oknie klikamy na *Next –>*. W kolejnym wprowadzamy ścieżkę, pod którą ma zostać zapisany projekt, oraz jego nazwę i nazwę głównej jednostki (**rysunek 1**). Pierwsza wartość jest dowolna, w kolejnych wpiszmy dla celów tego projektu *vga_oscilloscope*. W kolejnym oknie wybieramy *Empty project* i klikamy *Next –>* w tym i w kolejnym oknie (nie dodajemy żadnych istniejących plików). Następne okno umożliwia wybór układu z którym pracujemy. Wybieramy rodzinę *MAX 10* a następnie w polu *Name filter* wpisujemy nazwę naszego układu FPGA: *10M08DAF256C8GES* (**rysunek 2**). Wartości w kolejnym oknie pozostawiamy bez zmian, klikamy *Next –>*, ostatnie okno powinno mieć zawartość taką, jak pokazano na **rysunku 3**. Klikamy *Finish*. Projekt został właśnie utworzony.

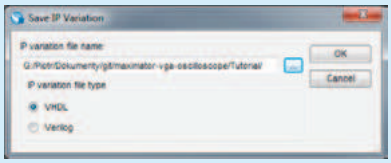
Elementy IP Cores – Pętla PLL oraz pamięć RAM

Na początku warto zacząć od elementów, które są już przygotowane przez osoby trzecie – tzw. *IP Cores*. Wśród nich są m. in. pętla PLL oraz pamięć RAM.

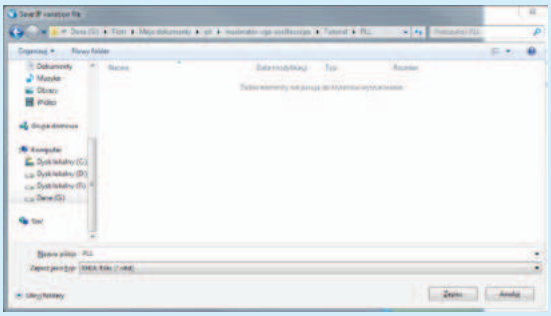
Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 22086, pass: 218655ee



Rysunek 4. Wyszukiwarka IP Cores



Rysunek 5. Wybór języka do generowania plików konfiguracyjnych pętli PLL



Rysunek 6. Wybór lokalizacji do zapisania plików konfiguracyjnych petli PLL

Pętla PLL

W wyszukiwarce *IP Catalog* (po prawej stronie ekranu, jeśli panel nie jest widoczny klikamy *View –>Utility windows –>IP Catalog*) wpisujemy *PLL* a następnie wybieramy *ALTPLL* za pomocą podwójnego kliknięcia (**rysunek 4**). W kolejnym oknie zaznaczamy *VHDL* oraz klikamy na „…”, aby wybrać, w którym miejscu zostanie utworzony plik z wygenerowanym wariantem konfiguracji pętli PLL (**rysunek 5**).

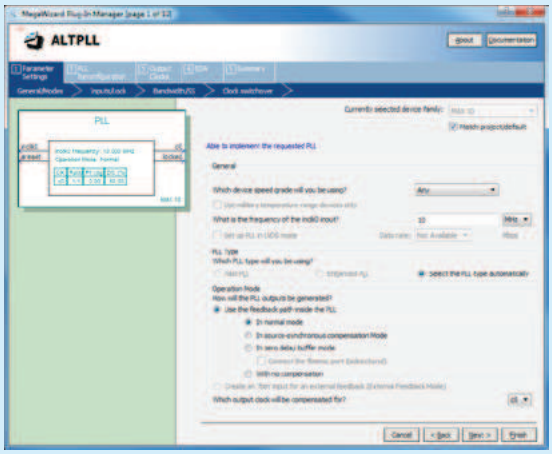
Polecam stworzenie osobnego folderu o nazwie *PLL* i nadanie naszemu plikowi nazwy *PLL.vhd* (**rysunek 6**).

Klikamy *Zapisz/Save* i następnie *OK*. Pojawia się okno konfiguracji pętli PLL. Ustawiamy wszystkie parametry jak na rysunku poniżej. W zasadzie wszystkie pozostawiamy w domyślnym ustawieniu, poza wpisaniem częstotliwości wejściowej jako 10MHz (*What is frequency of the inclk0 input?*). Okno nastaw pokazano na **rysunku 7**.

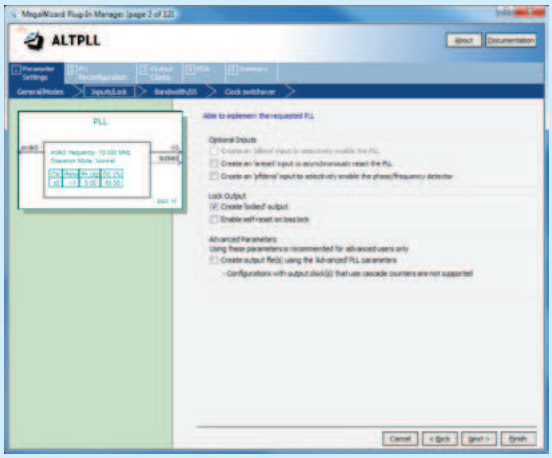
Klikamy *Next –>*. W kolejnym oknie (**rysunek 8**) odznaczamy *Create an 'assert' input…* Następnie klikamy *Next –>* do momentu aż wyświetli się nam menu konfiguracji *Output Clocks/clk c0*. Zaznaczamy *Use this clock* oraz *Enter output clock frequency* i wpisujemy 10 MHz jako *Requested Settings*. Klikamy *Next –>* (**rysunek 9**).

W kolejnym oknie (konfiguracja *c1*) zaznaczamy takie same opcje, z tym, że podajemy częstotliwość 50 MHz. Dla *c2* podajemy częstotliwość 65 MHz. Następnie, dwukrotnie klikamy *Finish* (**rysunek 10**).

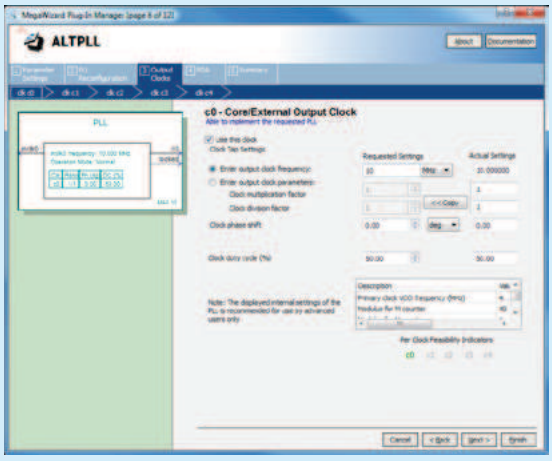
Podsumowując – dokonaliśmy generacji wariantu konfigurującego pętlę PLL w taki sposób, że dla wejściowej częstotliwości 10 MHz na wyjściu generuje



Rysunek 7. Okno właściwości pętli PLL



Rysunek 8. Ustawienie właściwości pętli PLL – krok 1



Rysunek 9. Ustawienie właściwości pętli PLL – krok 2

częstotliwości 10, 50 oraz 65 MHz. Dodatkowo, wyjście *locked* będzie informowało nas o tym, że pętla poprawnie zsynchronizowała się z sygnałem wejściowym – będzie to potrzebne dla modułu ADC.

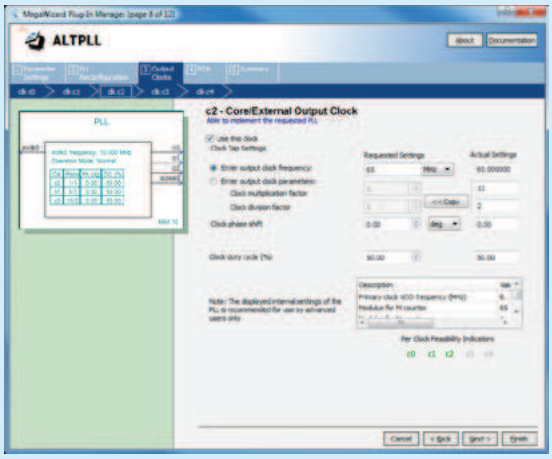
Pamięć RAM

Teraz kolej na konfigurację pamięci 2-portowej RAM. Rozpoczynamy jak poprzednio wyszukując w *IP Catalog* nazwę *RAM: 2-PORT*. Po jej wybraniu przez podwójne kliknięcie podobnie jak poprzednio wybieramy język *VHDL* oraz pliki zapiszemy w folderze *RAM_DISPLAY* a plik wariantu będzie miał nazwę *RAM_DISPLAY.vhd*. Po pokazaniu się okna konfiguratora, pozostawiamy

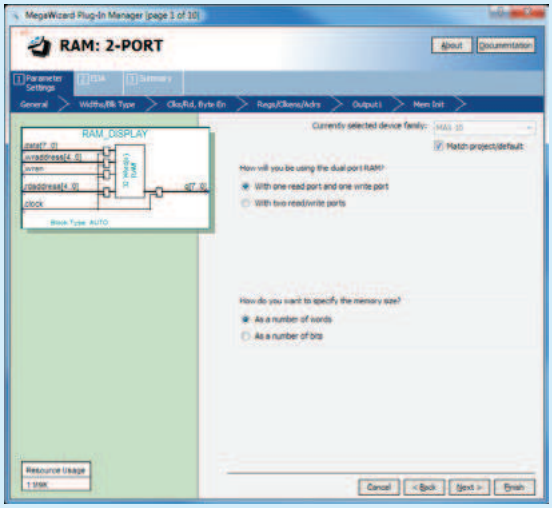
Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 22086, pass: 218655ee

wartości domyślne, aby pamięć miała jeden port do zapisu i jeden do odczytu (**rysunek 11**). W kolejnym oknie ustawiamy rozmiar pamięci na 1024 słowa oraz szerokość słowa na 9 bitów (**rysunek 12**).

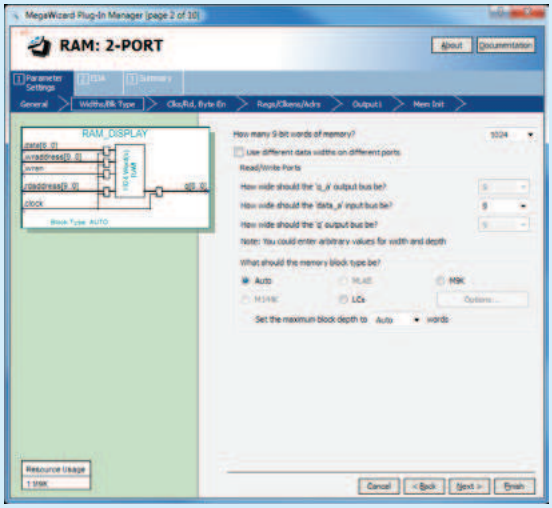
Skąd takie wartości? Jak już wcześniej wspomniałem skorzystamy z wyświetlania obrazu o rozdzielczości 1024×768 punktów. Zatem będziemy mogli w szerokości ekranu zmieścić 1024 próbki (stąd ilość słów w pamięci).



Rysunek 10. Poprawnie skonfigurowana pętla PLL



Rysunek 11. Okno właściwości pamięci



Rysunek 12. Ustawienie wielkości pamięci na 1024 słowa oraz szerokości słowa na 9 bitów

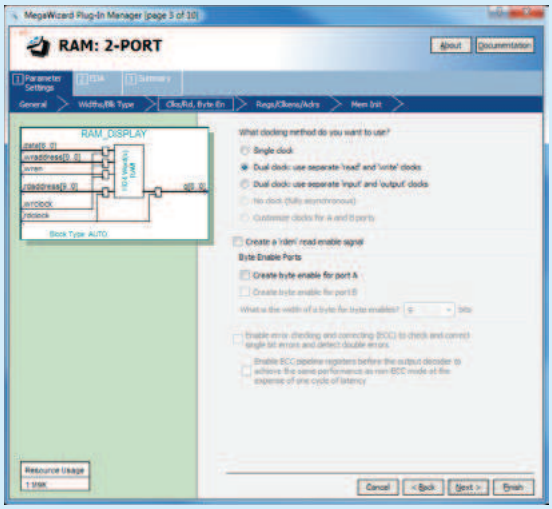
Z kolei wybieramy 9 bitów jako szerokość słowa, gdyż tyle bitów wyniku pomiaru będziemy prezentować. Wynika to z 2 powodów. Po pierwsze dane są dosyć zaszumione i 8–9 bitów bez uśredniania wyników to rozsądna wartość. Po drugie 9 bitów daje nam 512 różnych wartości, czyli prawie 768 pikseli – wypełnimy dzięki temu bez konieczności przeliczania prawie cały ekran.

W kolejnym oknie zaznaczamy *Dual clock: use separate 'read' and 'write' clocks*. Dzięki temu pamięć będzie miała 2 wejścia zegarowe – jedno do zapisu, drugie do odczytu – zatem zapis I odczyt będą w 100% niezależne (**rysunek 13**).

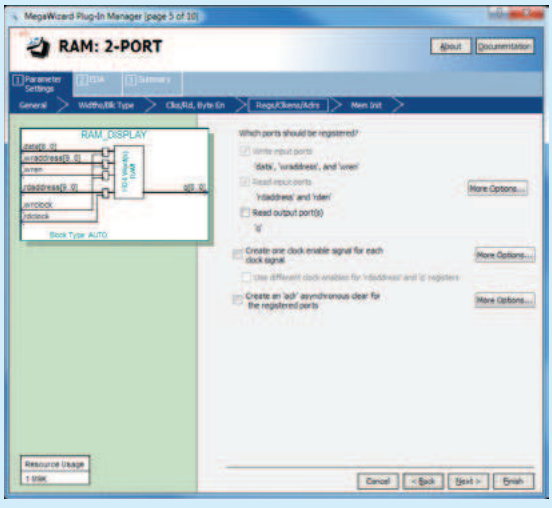
Następnie odznaczamy opcję *Read output port(s) 'q'*. Dzięki temu nie będzie generowany zatrzask na wyjściu z pamięci (**rysunek 14**). Klikamy następnie 2 krotnie *Finish*. Stworzyliśmy właśnie 2 portową pamięć RAM mieszczącą 1024 słowa 9-bitowe z w pełni rozdzielonym zapisem i odczytem.

Przetwornik ADC – serce oscyloskopu

Teraz przyszedł czas na konfigurację przetwornika ADC – tym razem mamy nieco więcej pracy, gdyż domyślnie przetwornik ten jest przeznaczony do pracy w systemie mikroprocesorowym budowanym za pomocą QSYS. Rozpoczynamy od wybrania *Tools –>Qsys*. Następnie usuniemy domyślnie wstawiony moduł połączenia zegara

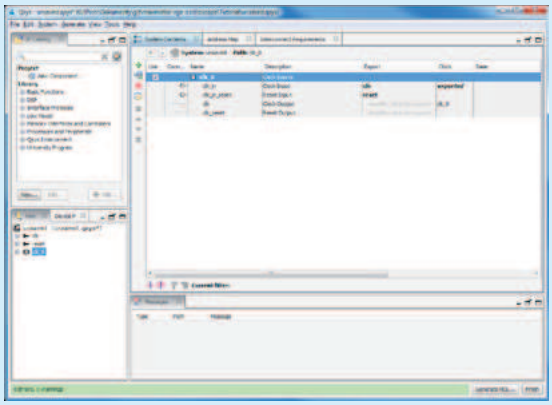


Rysunek 13. Zaznaczenie *Dual clock: use separate 'read' and 'write' clocks*



Rysunek 14. Odznaczenie opcji *Read output port(s) 'q'*

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 22086, pass: 218655ee



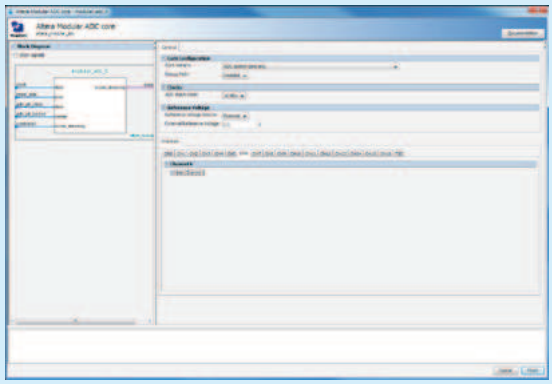
Rysunek 15. Okno QSYS z domyślnymi komponentami

systemu. Robimy to klikając na jego nazwę a następnie wciskając na klawiaturze *Delete* lub czerwony krzyżyk po lewej stronie głównego okna (**rysunek 15**). Następnie w wyszukiwarce *IP Catalog* wpisujemy *Altera Modular ADC core* i wybieramy podwójnym kliknięciem moduł o takiej nazwie. W oknie, które się pokaże wybieramy *Core Variant: ADC control core only*. Widzimy tu także ustawienie częstotliwości taktującej przetwornik na 10 MHz. Następnie wybieramy zakładkę *CH6* i zaznaczamy *Use Channel 6* (**rysunek 16**).

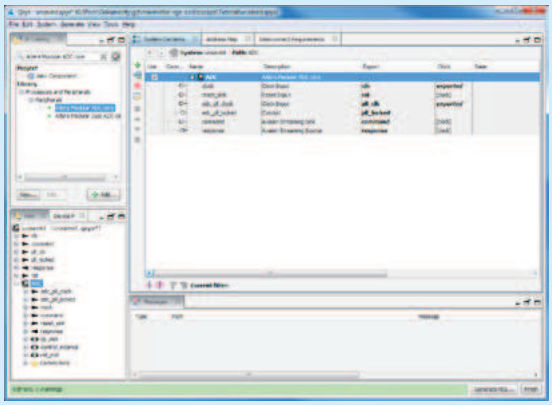
Po kliknięciu na *Finish* ustawienia zostaną zapisane. Konfiguracja taka daje nam bezpośredni dostęp do sterowania przetwornikiem ADC. Dodatkowo skonfigurowaliśmy przetwornik, aby używał kanału 6 do pomiaru. Kanał 6 przetwornika jest podłączony do wyprowadzenia ANIN5 (A5) na płytce.

Teraz, jako że nie będziemy korzystać z możliwości łączenia przetwornika do innych elementów systemu (jest on wszak jedynym elementem naszego „systemu"), musimy dokonać wyeksportowania jego wyprowadzeń, aby były one widoczne poza modułem wygenerowanym przez *Qsys*. Aby tego dokonać klikamy dwukrotnie na kolejne pola w kolumnie *Export* nadając nazwy kolejnym wyprowadzeniom: *clk, rst, pll_clk, pll_locked, command, response*. Dodatkowo klikając prawym klawiszem myszy na polu modułu możemy wybrać opcję *Rename* i zmienić jego nazwę na *ADC*. Po tych zabiegach okno powinno wyglądać mniej więcej tak, jak na **rysunku 17**.

Wybieramy *File –> Save As…* i zapisujemy plik w nowym folderze *qsys* pod nazwą *ADC*. Powinno pojawić się okienko *Save System: completed successfully*. Klikamy w nim *Close*. Przydatną opcją jest *Generate –> Instantion Template*. Tam po wybraniu *HDL Language: VHDL* możemy skopiować wzór deklaracji komponentu



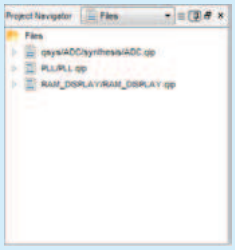
Rysunek 16. Wybranie kanału 6 przetwornika ADC



Rysunek 17. Okno QSYS

i jego instancji. Jest to szczególnie przydatne, jeśli sami piszemy kod.

Aby wygenerować pliki klikamy na *Generate HDL…* W wyświetlonym oknie wybieramy *Create HDL design files for synthesis: VHDL*. Resztę opcji pozostawiamy bez zmian, klikamy *Generate*. Jeśli wszystko poszło zgodnie z planem powinniśmy otrzymać komunikat *Generate: completed successfully*.



Rysunek 18. Nawigator projektu – widok plików (*Files*)

Po kliknięciu *Close* i *Finish* klikamy także *OK* w oknie, które infomruje o konieczności ręcznego dodania wygenerowanego systemu do projektu. Klikamy na *Project –> Add/Remove Files in Project…*, w oknie klikamy na … obok okna *File name* i wybieramy plik <ścieżka_projektu>*qsys**ADC**synthesis**ADC.qip*, po czym klikamy na *Add* i *OK*. Teraz w okienku *Project Navigator* (domyślnie lewy górny róg, jeśli nie widoczne – patrz *View –>…*) wybierzmy widok *Files*. Powinniśmy widzieć obraz, jak na **rysunku 18**.

Moduły tworzone w VHDL

Teraz już wyeksploatowaliśmy możliwości automatycznego generowania bloków przez środowisko Quartus – czas na trochę prawdziwej pracy i zaprojektowanie logiki działania naszego oscyloskopu.

ADC Command (dodatkowe informacje na FTP-ie EP)

To chyba jeden z prostszych bloków do zaprojektowania. Jego zadanie to ustawienie na liniach sterujących ADC kanału, z którego ma zostać pobrana próbka (u nas kanał 6 – jeśli wybralibyśmy w konfiguracji ADC także inne kanały moglibyśmy teraz zmieniając tą wartość pobierać próbki napięcia z innych kanałów) oraz takiej kombinacji stanów, aby za każdym cyklem zegarowym komenda ta została wysłana do przetwornika, oraz zatrzaśnięcie rezultatu pomiaru w momencie, kiedy zostanie on zaprezentowany na wyjściu przetwornika. Aby dodać nowy plik do projektu klikamy: *File –> New…* następnie *Design Files –> VHDL File* i klikamy *OK*. W treści wpisujemy: (adc_command.vhd, **rysunek 19**) Widzimy tu wybranie kanału 6 (binarnie 00110), ustawienie pozostałych linii interfejsu w odpowiednie stany oraz proces zatrzaskujący wynik pomiaru. Szczegółowe informacje znajdziemy w dokumentacji modułu ADC.

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 22086, pass: 218655ee

```
-- Project:       maximator-vga-oscilloscope
-- File:          adc_command.vhd
-- Version:       1.0 (13.05.2016)
-- Author:        Piotr Rzeszut (http://piotr94.net21.pl)
-- Description:   Generates command signals for ADC core and latches conversion result

library ieee ;

use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity adc_command is
    port(
        clk                 : in  std_logic;
        response_valid      : in  std_logic;
        response_channel    : in  std_logic_vector(5 downto 0);
        response_data       : in  std_logic_vector(11 downto 0);
        response_startofpacket : in  std_logic;
        response_endofpacket : in  std_logic;
        command_channel     : out std_logic_vector(5 downto 0);
        command_valid       : out std_logic;
        command_startofpacket : out std_logic;
        command_endofpacket : out std_logic;
        command_ready       : out std_logic;
        valid_data          : out std_logic_vector(11 downto 0)
    );
end adc_command;

architecture basic of adc_command is
begin
    command_channel     <= "00110";
    command_valid       <= '1';
    command_startofpacket <= '1';
    command_endofpacket <= '1';
    command_ready       <= '1';

    process (clk) is
    begin
        if rising_edge(clk) and response_valid = '1' then
            valid_data <= response_data;
        end if;
    end process;

end basic;
```

Rysunek 19. Zmodyfikowany program

Aby zapisać plik klikamy *File –> Save…* Polecam znów utworzenie folderu src, a następnie zapisanie pliku pod nazwą taką, jak nazwa *entity*, który zawiera. Plik zostanie automatycznie dodany do projektu.

VGA Driver

Jako następny warto zaprojektować moduł sterowania interfejsem VGA. Na początku jednak przyjrzyjmy się temu, jak jest generowany obraz VGA. Interfejs VGA składa się z 5 aktywnych linii:

- **R** – analogowe wejście – nasycenie koloru czerwonego,
- **G** – analogowe wejście – nasycenie koloru zielonego,
- **B** – analogowe wejście – nasycenie koloru niebieskiego,
- **HSYNC** – cyfrowe wejście – synchronizacja pozioma,
- **VSYNC** – cyfrowe wejście – synchronizacja pionowa.

Podstawowym elementem obrazu jest linia (**rysunek 20**), zaś czas jest tu odmierzany poprzez liczbę pikseli (czas trwania piksela jest zależny od częstotliwości, z jaką są one nadawane). Generowanie linii i jej omawianie można rozpocząć od dowolnego miejsca, zatem możemy zacząć na przykład od początku nadawania obrazu (display, PIXEL DATA). W tym momencie na liniach RGB należy zmieniać z częstotliwością nadawania pikseli wartości napięć, tak, aby odpowiadały one pożądanemu nasyceniu każdej ze składowych koloru. W naszym wypadku linie te podłączone są do cyfrowych wyjść układu FPGA, zatem możemy ustawić je w 2 stanach – 100% nasycenia koloru, lub 0%.

Następnie występuje czas Front Porch – piksele niewyświetlane, poza obszarem wyświetlania. W tym czasie linie RGB powinny być ustawione na wyświetlanie koloru czarnego. Kolejny element linii to impuls synchronizacyjny (aktywny w stanie niskim). Po nim następuje czas Back Porch – znów czarne piksele, które nie będą wyświetlane a znajdują się przed początkiem linii.

Czasy Front i Back Porch miały szczególne znaczenie przy monitorach kineskopowych, gdzie ustawienie wiązki elektronów na początku nowej linii wymagało pewnego czasu. Oczywiście cały obraz składa się z wielu linii, zatem tym razem naszą podstawową jednostką czasu będzie linia (**rysunek 21**). Tu sytuacja jest analogiczna jak w przypadku pojedynczej linii, z tym, że w czasie trwania synchronizacji i okresów back/front porch linia musi cały czas „przybierać" kolor czarny.

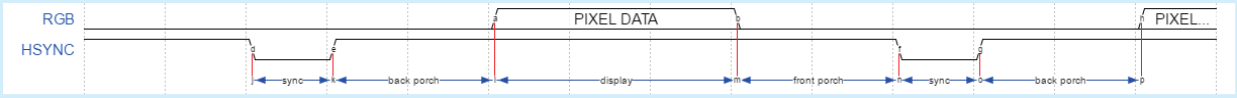
Po tym krótkim omówieniu zapewne każdy z łatwością zaprojektowałby odpowiedni sterownik. Rzecz jasna poszczególne czasy oraz częstotliwość muszą mieć ściśle określone wartości, które z łatwością znajdziemy w Internecie. Moja implementacja tego drivera dodatkowo umożliwia dostosowanie się do konkretnej rozdzielczości za pomocą parametrów.

Moduł ten na wyjściu generuje 5 wspomnianych sygnałów VGA. Dodatkowo zlicza kolejne piksele wyświetlone na ekranie (*px_addr_int*) począwszy od lewego górnego rogu, liniami aż do prawego dolnego rogu ekranu. Moduł cały czas asynchronicznie przelicza ten adres na numer wiersza i kolumny, w której tenże piksel się znajduje. Moduł na wejściu, oprócz sygnału zegarowego przyjmuje też dane o kolorze piksela(*vga_in*), którego adres jest aktualnie podawany na wyjściach *px_addr_*.... Moduł jest taktowany zboczem opadającym, dlatego przy współpracy z modułem taktowanym zboczem narastającym zachowana będzie odpowiednia sekwencja zatrzaskiwania danych o pikselu pochodzących z modułu generowania obrazu (u nas rysowania linii) oraz ustawiania dla tego modułu danych o adresie kolejnego piksela (**rysunek 22**).

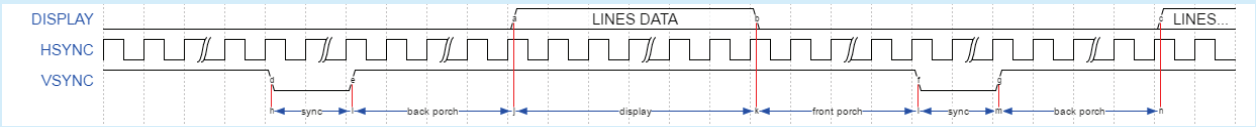
Line Drawer

Kolejnym ważnym elementem systemu będzie przygotowanie modułu rysującego linie. Moduł ten jak wspomnieliśmy będzie łączyć kolejne punkty na wykresie linią.

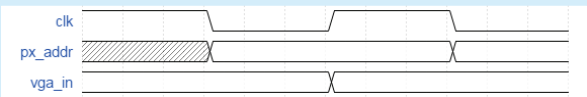
Wiemy, że skanowanie obrazu na ekranie odbywa się liniami, zatem każdy kolejny punkt w danej linii będzie odpowiadał kolejnej próbce danych. Stąd też od razu wniosek, że na wejściu moduł za każdym zboczem sygnału zegarowego powinien otrzymać wartość kolejnej próbki.



Rysunek 20. Podstawowy element obrazu – linia



Rysunek 21. Obraz złożony z linii wraz z sygnałami sterującymi.

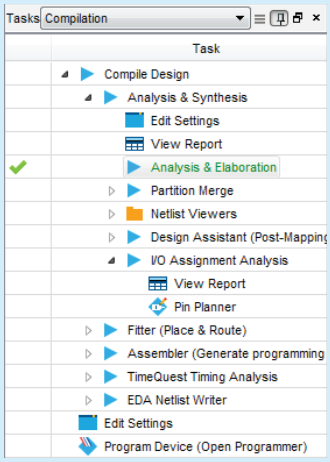


Rysunek 22. Sekwencja sygnałów kontrolnych modułu VGA

Moduł musi też pamiętać wartość próbki poprzedniej, gdyż narysowanie linii między kolejnymi próbkami znajdującymi się w kolejnych kolumnach będzie wymagało zaledwie sprawdzenia czy numer wiersza na ekranie znajduje

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 22086, pass: 218655ee

się pomiędzy tymi wartościami, czy też nie i w zależności od tego ustawienie koloru tła bądź linii. Dodatkowo w module przewidziano wejście do ustawiania położenia wykresu na ekranie (aby nie znajdował się on przy brzegach – jak pamiętamy wykorzystamy 512 z dostępnych 768 pikseli) oraz wyjście *px_req*, które informuje o tym, czy moduł dany piksel wykorzystuje do rysowania wykresu, czy też tylko „rysuje tło". Funkcja ta może być przydatna w przypadku chęci nałożenia na wykres np. siatki, czy innych informacji. Ponadto odwracamy dane względem osi X (im wyższe napięcie – tym wyższa liczba, im wyższy numer linii – tym niżej ta linia położona, zatem jest konieczne odwrócenie danych).



Rysunek 23. Okno z listą zadań, przebiegu procesu kompilacji

Trigger

Ostatnim komponentem systemu będzie układ wyzwalający. Jak już wcześniej wspomnieliśmy ma on powodować zapisywanie próbek z przetwornika ADC do pamięci RAM. Musimy zatem po wykryciu wciśnięcia przycisku rozpocząć zapis od pierwszego adresu w pamięci ram, aż do jej zapełnienia. Sterujemy zatem adresem pamięci RAM na porcie do zapisu oraz sygnałem włączenia zapisu (*wr_en*). Dodatkowo moduł zawiera także dodatkowy licznik, dzięki czemu zapisujemy próbki rzadziej, niż są one pobierane przez przetwornik.

Moduł TOP, czyli połączenie klocków w działający układ

Teraz przyszła kolej na zwieńczenie naszej pracy i połączenie wszystkich modułów ze sobą. I tak łączymy:

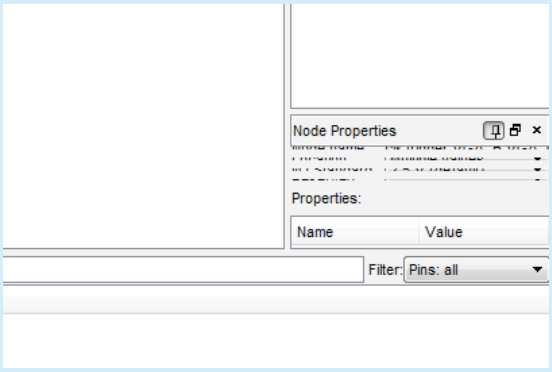
1. Sygnały zegarowe odpowiednich modułów oraz pętli PLL:
   b. 10 MHz z pętli PLL do taktowania przetwornika ADC, wraz z sygnałem *locked*
   c. 10 MHz do taktowania sterownika ADC (wejście *clk_clk*)
   d. 10 MHz do taktowania układu *adc_command*
   e. 65 MHz do taktowania sterownika VGA, układu rysującego linię oraz do taktowania odczytu z pamięci RAM.
   f. 50 MHz do taktowania modułu wyzwalania oraz zapisu do pamięci RAM
2. Sygnały sterujące między przetwornikiem ADC i modułem zlecającym konwersję *adc_command*.
3. Wyjście bloku *adc_command* z zatrzaśniętym ostatnim wynikiem pomiaru (*valid_data*) z portem do zapisu pamięci RAM (zapis zostanie dokonany tylko, jeśli moduł wyzwalania ustawi odpowiednio linię *wr_en)*. Tu także pomijamy 3 najmniej znaczące bity z wyniku pomiaru.
4. Wyjścia bloku wyzwalającego (adres i zezwolenie na zapis) z pamięcią RAM.
5. Adres odczytu RAM z numerem kolumny ze sterownika VGA
6. Wyjście pamięci RAM z wejściem danych układu rysującego linię
7. Numer wiersza ze sterownika VGA z wejściem układu rysującego linię
8. Wyjście układu rysującego linię z wejściem informacji o kolorze kolejnego piksela

Konfigurowanie pinów i synteza

Na tym etapie nasz projekt jest gotowy do syntezy. Jedyne, czego nam brak, to przekazania syntezerowi informacji o użytych przez nas wyprowadzeniach. Aby tego dokonać musimy najpierw kliknąć dwukrotnie w oknie *Tasks* z wybranym widokiem *Compilation* na *Analysis & Elaboration*. Jeśli wszystko przebiegło bez problemów powinniśmy po pewnym czasie zobaczyć widok, jak na **rysunku 23**. W przypadku wystąpienia jakichkolwiek błędów wyświetlą się one w konsoli.

Po prawidłowej analizie możemy przejść do menu przypisywania pinów, w tym celu wybieramy *Assignments –>Pin Planner*. Spowoduje to otwarcie narzędzia do przypisywania pinów, w którym widzimy widok wyprowadzeń układu FPGA (**rysunek 24**). W tabeli na dole widzimy nazwy naszych wyprowadzeń użyte w głównym module projektu, ich kierunek. Teraz pora na uzupełnienie pola *Location* – czyli nazwy pinu. Zaczynając od *clk* wpisujemy kolejno: *L3, B10, M1,*

Rysunek 24. Okna narzędzia do przyporządkowywania wyprowadzeń



Rysunek 25. Czasami panel [i]Node Properties[/i] zostaje wyświetlony jako zwinięty – wystarczy go odpowiednio rozciągnąć.

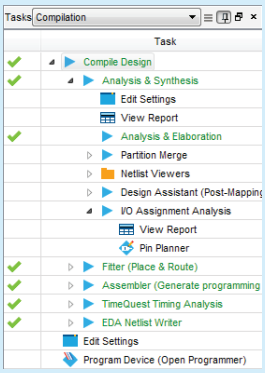
Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 22086, pass: 218655ee

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
clk	Input	PIN_L3	2	B2_N0	3.3-V LVTTL		8mA (default)			
trigger	Input	PIN_B10	8	B8_N0	3.3-V LVTTL		8mA (default)			
VGA_B	Output	PIN_M1	2	B2_N0	3.3-V LVTTL		8mA (default)	2 (default)		
VGA_G	Output	PIN_N1	2	B2_N0	3.3-V LVTTL		8mA (default)	2 (default)		
VGA_HS	Output	PIN_L1	2	B2_N0	3.3-V LVTTL		8mA (default)	2 (default)		
VGA_R	Output	PIN_R1	3	B3_N0	3.3-V LVTTL		8mA (default)	2 (default)		
VGA_VS	Output	PIN_J1	1B	B1_N0	3.3-V LVTTL		8mA (default)	2 (default)		
<<new node>>										

Rysunek 26. Wygląd uzupełnionej listy wyprowadzeń

N1, L1, R1, J1. Są to piny zgodne z połączeniami na płytce MAXimator (patrz: dokumentacja). Warto zauważyć, że program sam uzupełnia przedrostek *PIN_*.

Następnie warto dobrać standard wyprowadzeń. Możemy dla każdego pinu ręcznie zmienić go na *3.3-V LVTTL*, lub zaznaczyć wszystkie piny (klikając na nazwy w kolumnie *Node Name* z *CTRL* lub *SHIFT*) Następnie klikamy na *View –>Node Propertis*. Czasem panel ten pojawi się po prawej stronie okna zwinięty – wystarczy go odpowiednio rozciągnąć i wybrać *I/O Standard: 3.3-V LVTTL* (**rysunek 25**).



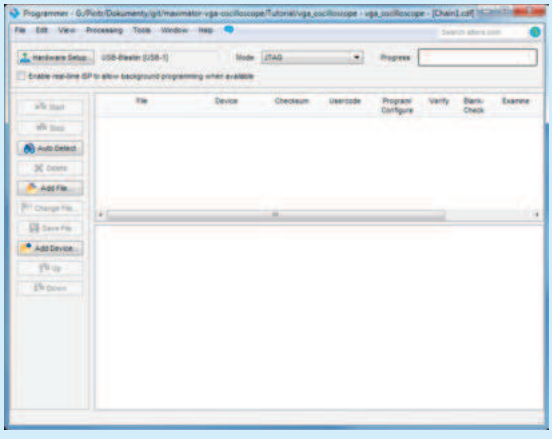
Rysunek 27. Okno z listą zadań

Resztę konfiguracji zostawiamy bez zmian. Lista powinna wyglądać na tym etapie jak na **rysunku 26**. o zamknięciu okna klikamy w *Tasks: Compilation* na *Compile Design*. Po pewnym czasie w przypadku braku błędów zobaczymy okienko, jak na **rysunku 27**. Nasz oscyloskop został poprawnie zsyntezowany – czas na wgranie konfiguracji i testy.

Programowanie i testy

W celu zaprogramowania układu klikamy 2-krotnie na *Program Device (Open Programmer)* – **rysunek 28**. Oczywiście, musimy podpiąć programator do płytki i komputera oraz zasilić płytkę. Jeśli obok *Hardware Setup…* nie wyświetla się nazwa programatora to musimy kliknąć na ten przycisk i z listy rozwijanej wybrać odpowiedni programator.

Następnie klikamy na *Auto Detect*. Pojawia się okno z prośbą o sprecyzowanie układu, pozostawiamy w nim zaznaczenie przy *10M08DA*. Klikamy na *OK*. Na liście pojawia się nasz układ z wpisem *<none>* w kolumnie plik. Klikamy 2-krotnie na to miejsce i wybieramy *./output_files/vga_oscilloscope.sof*, aby wgrać konfigurację tymczasową do pamięci ram. Wgranie pliku pof spowodowałoby zapisanie naszej konfiguracji w pamięci flash układu FPGA. Następnie zaznaczamy pole w kolumnie *Program/Configure* i klikamy na *Start*. Powinniśmy zobaczyć w prawym górnym rogu informację o prawidłowym przebiegu operacji (**rysunek 29**).



Rysunek 28. Okno programatora

Testy oscyloskopu

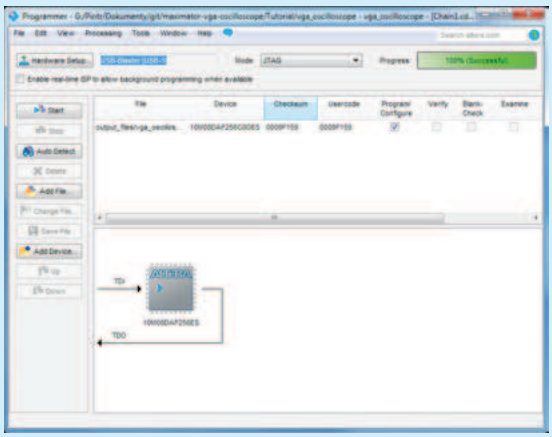
Teraz już czas na podłączenie monitora VGA i przetestowanie naszego projektu. Na wejście *ANIN5* podajemy sygnał z generatora o napięciach z zakresu 0…2,5 V. Musimy tu zachować szczególną ostrożność, aby, o ile generator ma takie możliwości, nie przekroczyć tego zakresu napięć (wejścia są chronione, tak, aby wytrzymać ewentualne nieco wyższe i nieco niższe napięcia, jednak lepiej nie wystawiać tych zabezpieczeń na próbę). Jeśli nie posiadamy profesjonalnego generatora możemy użyć zewnętrznego potencjometru (ten na płytce nie będzie działać bez modyfikacji ustawień bloku ADC i kodu – należy wtedy wybrać kanał *CH0* i odpowiednio żądać konwersji na tym kanale w bloku sterującym przetwornikiem)

Częstotliwość ustawiamy na dosyć niską, z zakresu 2–10 Hz (nasz układ próbkuje 500 razy na sekundę – zatem na ekranie o 1024 punktach w poziomie zaobserwujemy nieco ponad 2 sekundy przebiegu).

Aby wyzwolić próbkowanie klikamy przycisk *DEV_CLR* na płytce. Powinniśmy zobaczyć na ekranie odwzorowanie naszego przebiegu.

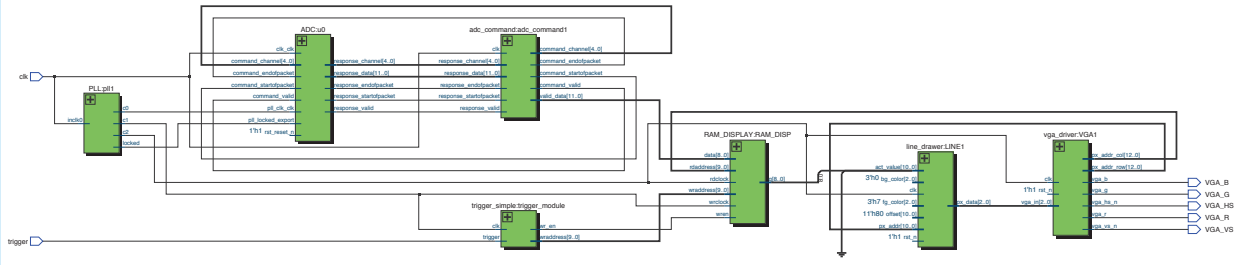
Uzupełnienie funkcjonalności

Widok kompletnego projektu pokazano na **rysunku 30**. Największą wadą naszego oscyloskopu jest brak wyzwalania związanego z sygnałem, przez co próbkowanie ciągłe nie jest możliwe – za każdym razem otrzymamy przebieg inaczej przesunięty w czasie. Ponadto nie demonstrowaliśmy jeszcze wykorzystania nakładania obrazów. W dodatkowym kodzie zbudowano 3 kolejne moduły:



Rysunek 29. Okno programatora – komunikat wyświetlany po poprawnej konfiguracji układu

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 22086, pass: 218655ee



Rysunek 30. Widok kompletnego projektu

- *trigger_drawer* – rysujący linię poziomą na poziomie progu wyzwolenia
- *priority_mux* – kontrolujący nakładanie się obrazu (tu w zasadzie sprawdzający tylko, czy moduł rysujący przebieg chce rysować przebieg)
- *trigger_hyst* – moduł implementujący komparator z histerezą – układ wyzwalający. Dzięki zastosowaniu histerezy w przypadku zaszumionego przebiegu nie nastąpi wyzwolenie np. na zaszumionym zboczu opadającym, mimo iż pożądane byłoby zbocze narastające. Moduł posiada dwa wyjścia informujące o wykryciu obu zboczy.

Ponadto, te moduły odpowiednio połączono w pliku głównym, oraz usunięto wejście dla przycisku wyzwalającego, natomiast w module *trigger_simple* zmieniono preskaler, aby próbkować z częstotliwością 100 kHz. Teraz po podaniu odpowiednio szybkiego przebiegu przechodzącego przez napięcie około 1,25 V i posiadającego odpowiednią amplitudę (np. sinus 1 V + 1,25 V DC, częstotliwość 10 kHz) zobaczymy ciągłe jego odświeżanie, bez naszego udziału.

Tym sposobem stworzyliśmy w oparciu o tylko jeden układ FPGA uproszczony modelowy oscyloskop, implementujący najważniejsze funkcje takich urządzeń, czyli próbkowanie z wyzwalaniem oraz wyświetlanie przebiegu.

Co dalej? Czyli zadanie domowe

Po zaprojektowaniu (celowo nie używam tu słowa programowanie, bo w VHDL'u opisujemy elektronikę, a nie piszemy klasyczny program jak na mikrokontroler!) takiego oscyloskopu z jednej strony powinniśmy czuć pewną satysfakcję z jego uruchomienia. Z drugiej jednak strony każdy uczący się powinien zawsze czuć niedosyt. Co zatem można jeszcze zrobić? Już spieszę z listą „TO-DO":

1. Zmiana zbocza wyzwalania
2. Regulacja poziomu wyzwalania
3. Regulacja histerezy
4. Rysowanie siatki
5. Regulacja częstotliwości próbkowania
6. Próbkowanie naprzemienne z 2 kanałów i jego konfiguracja

Zadania te można zrealizować z wykorzystaniem modułu MAXimator Expander – mamy na nim dostępny wyświetlacz 4-pozycyjny LED do prostego wyświetlania informacji. Sumarycznie do dyspozycji mamy też 4 przyciski (FPGA to nie „Arduino" – przycisk reset możemy dowolnie wykorzystać, podobnie jak w pierwszej wersji używaliśmy przycisku podpiętego do linii z dedykowaną funkcją nDEV_CLR jako normalnego wejścia!)

Zadania 1–4 są stosunkowo proste, zadanie 5 wymaga pewnych modyfikacji modułu *trigger_simple*, zaś zadanie 6 wymaga już lepszego zrozumienia działania systemu przetwornika i zmieniania kanału dla kolejnych pomiarów, oraz osobnego zatrzaskiwania wyników dla każdego z kanałów. Moduł rysowania linii i pamięci ram możemy zduplikować.

Kolejnym pomysłem może być zastosowanie do wprowadzania pewnych danych zamiast przycisków – enkoderów obrotowych.

Najbardziej ambitni mogą pomyśleć o wyświetlaniu tych wszystkich parametrów w formie tekstowej na ekranie VGA – tu już konieczne jest dobre przemyślenie całego systemu oraz wykonanie generatora czcionek.

Mam nadzieję, że te pomysły zainspirują Was do realizacji ciekawych projektów i dalszej nauki programowania układów FPGA (Wróć! Przecież to nie jest zwykłe programowanie!).

Piotr Rzeszut, AGH
http://piotr94.net21.pl

Autor artykułu składa podziękowania dla panów dr Pawła Rajdy i dr Jerzego Kasperka za opiekę w czasie realizacji projektu.

REKLAMA

Wydanie specjalne magazynu Gitarzysta wyciąga rękę do wszystkich tych, którzy chcą nagrywać muzykę w domu – a badania pokazują, że jest to obecnie niezwykle szeroka i dynamicznie rosnąca grupa odbiorców. Zaczynamy od opisania kompletnych podstaw, sposobów podłączania instrumentu, dostępnych programów do nagrywania muzyki, sprzętu niezbędnego do tego. Polecamy wybrane produkty z działów gitara elektryczna, gitara akustyczna, gitara basowa, perkusja. Omawiamy zasady nagrywania tych instrumentów, dzielimy się poradami zawodowców, podpowiadamy jak wykonać miks i mastering. Co więcej omawiamy także strukturę piosenki i tego jak powinien zostać napisany przebój. Na koniec kilka wskazówek dotyczących biznesu muzycznego i PR. Przewodnik Domowe Studio to pozycja obowiązkowa dla każdego!

PRZEJRZYSZ I KUPISZ NA WWW.ULUBIONYKIOSK.PL

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: ftp://ep.com.pl, user: 22086, pass: 218655ee