ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Sprzętowa implementacja algorytmów

9. Wykorzystanie pamięci blokowej RAM w układzie FPGA.

Lider projektu: Politechnika Warszawska

Autor: Łukasz Mik

Akademia Nauk Stosowanych w Tarnowie

GRANT NUMBER: 2020-1-PL01-KA226-HE-095653

Declaration

This laboratory instruction has been prepared in the context of the ENGINE project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

Funding Disclaimer

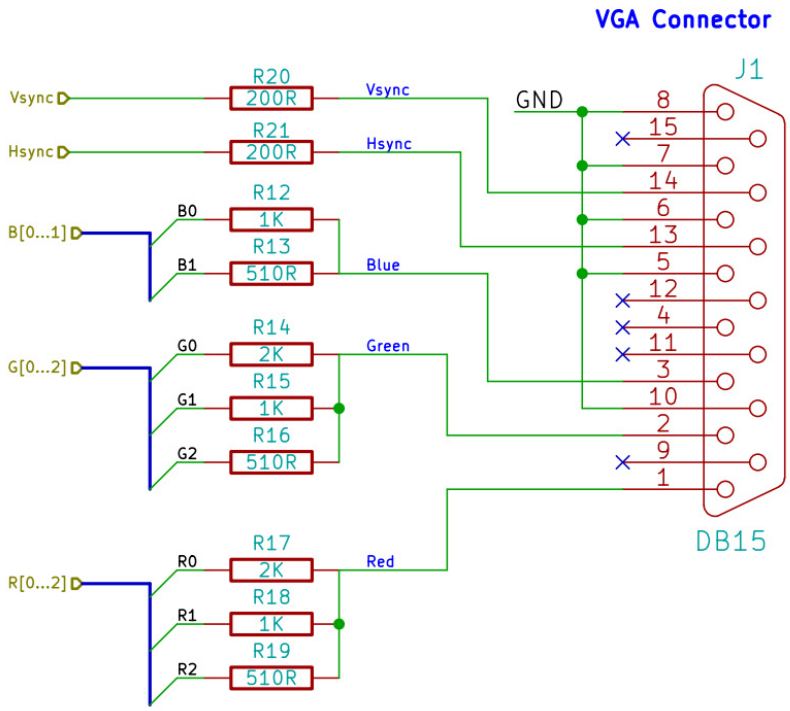
This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

I. Wyświetlanie obrazów zapisanych w pamięci blokowej RAM układu Spartan-3A, na ekranie monitora VGA.

Układ Spartan-3A, w który jest wyposażona płyta Numato Elbert V2 ma w swoich zasobach pamięć blokową RAM o łącznej pojemności 54 kb oraz pamięć rozproszoną (złożoną z konfigurowalnych bloków logicznych) o pojemności 11 kb. W trakcie zajęć zajmiemy się wykorzystaniem pamięci blokowej (BRAM).

Przed przystąpieniem do realizacji ćwiczenia, należy pobrać pliki źródłowe do projektu z folderu „VHDL Sources" i zapisać je w folderze o odpowiedniej nazwie.

Podczas zajęć zostanie wykorzystany port VGA, którego schemat połączeń z układem FPGA został przedstawiony na poniższym rysunku.



Widać na nim, że zawiera 3 proste przetworniki DAC w postaci drabinek rezystorów. Dla kolorów: czerwonego i zielonego jest to przetwornik 3 – bitowy, natomiast dla koloru niebieskiego tylko 2 – bitowy. W związku z tym, konieczna jest konwersja 8 – bitowych wartości składowych kolorów RGB (odczytanych z pliku BMP) na docelowe wartości, odpowiadające długościom wektorów bitowych dla przetworników DAC. Do ćwiczenia został przygotowany w środowisku Matlab skrypt o nazwie *image_converter.m*, którego zadaniem jest wczytanie pliku BMP, redukcja długości bitów

poszczególnych składowych RGB oraz zapisanie uzyskanego w ten sposób wektora danych do pliku o rozszerzeniu *.*coe*.

Skrypt w Matlabie generuje wektor danych w postaci ciągów binarnych, dlatego w pierwszym wierszu pliku *.*coe* jako podstawa systemu liczbowego podana jest liczba 2. W kolejnym wierszu rozpoczyna się wektor danych, każda wartość jest zapisywana w oddzielnym wierszu. Fragment pliku *.*coe* został przedstawiony poniżej.

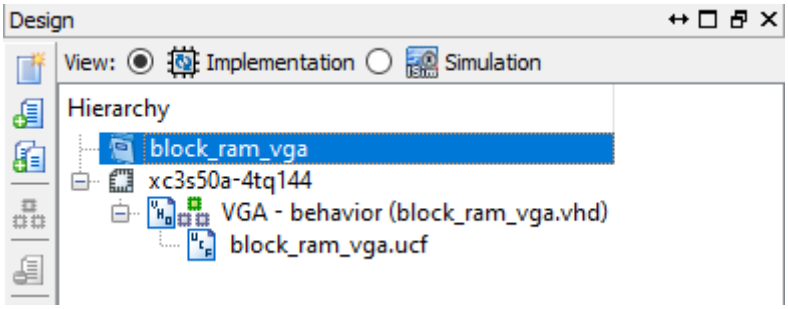
```
MEMORY_INITIALIZATION_RADIX=2;
MEMORY_INITIALIZATION_VECTOR=
01110010
01110010
01110010
01110010
01110010
01110010
01110010
01110010
01110010
01110010
01110010
.
.
.
```

KROK 1: Utworzenie nowego projektu, dodanie plików źródłowych oraz komponentu pamięci z wykorzystaniem generatora *IP Core*.

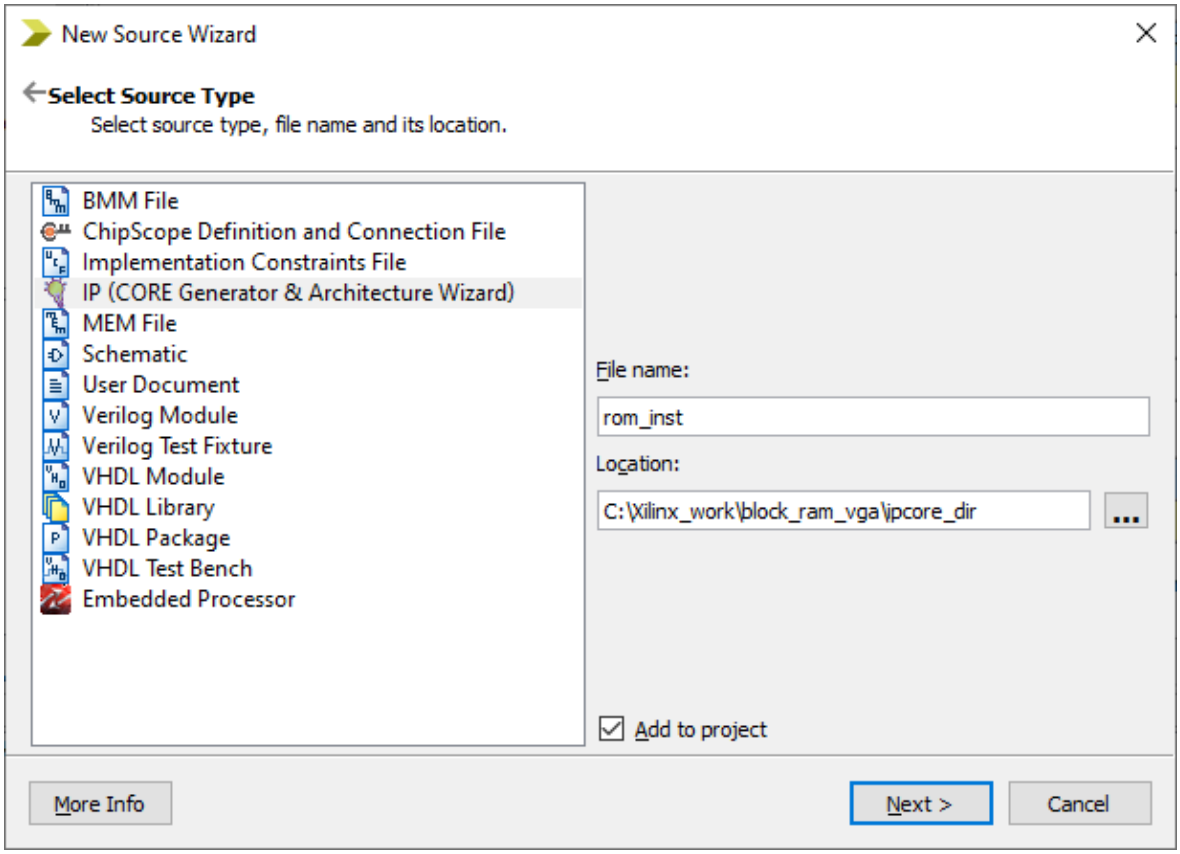
Uruchom program ISE Design Suite 14.7, z menu *File* wybierz opcję *New Project*. Wpisz odpowiednią nazwę projektu np. *block_ram_vga*, a w kolejnym oknie ustaw następujący układ docelowy dla tworzonego projektu:

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3A and Spartan3AN
Device	XC3S50A
Package	TQ144
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-200X
Enable Message Filtering	☐

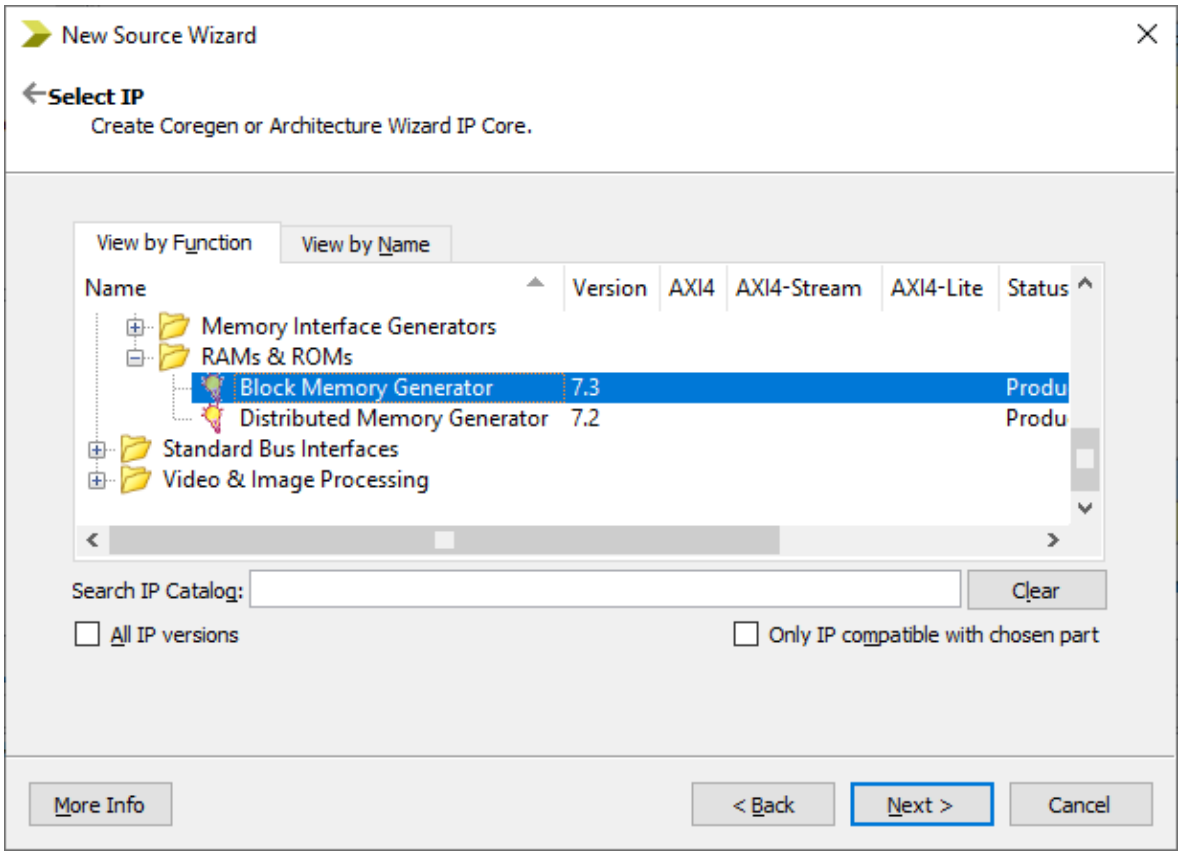
Po utworzeniu projektu i dodaniu pobranych wcześniej plików źródłowych w oknie *Design* powinny pojawić się wszystkie dodane pliki oraz typ wybranego układu:



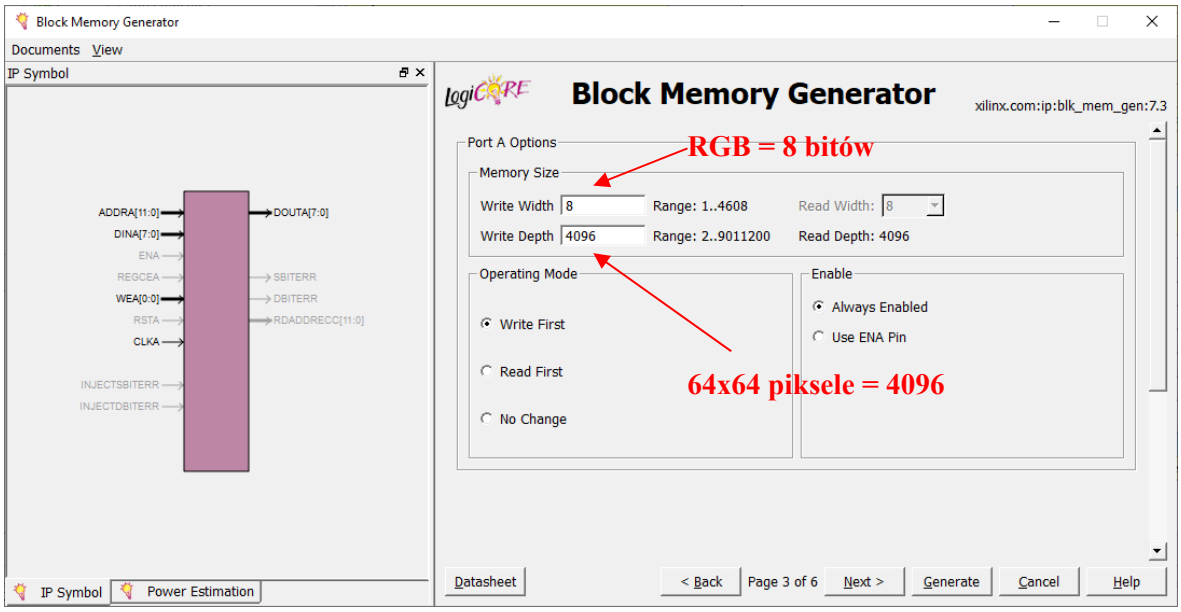
Z menu *Project* wybierz opcję *New Source*. W oknie, które się pojawi wybierz moduł IP (*Coregen & Architecture Wizard*). Jako nazwę wpisz *rom_inst*.



W kolejnym oknie rozwiń grupę *Memories & Storage Elements -> RAMs & ROMs -> Block Memory Geneerator 7.3*.

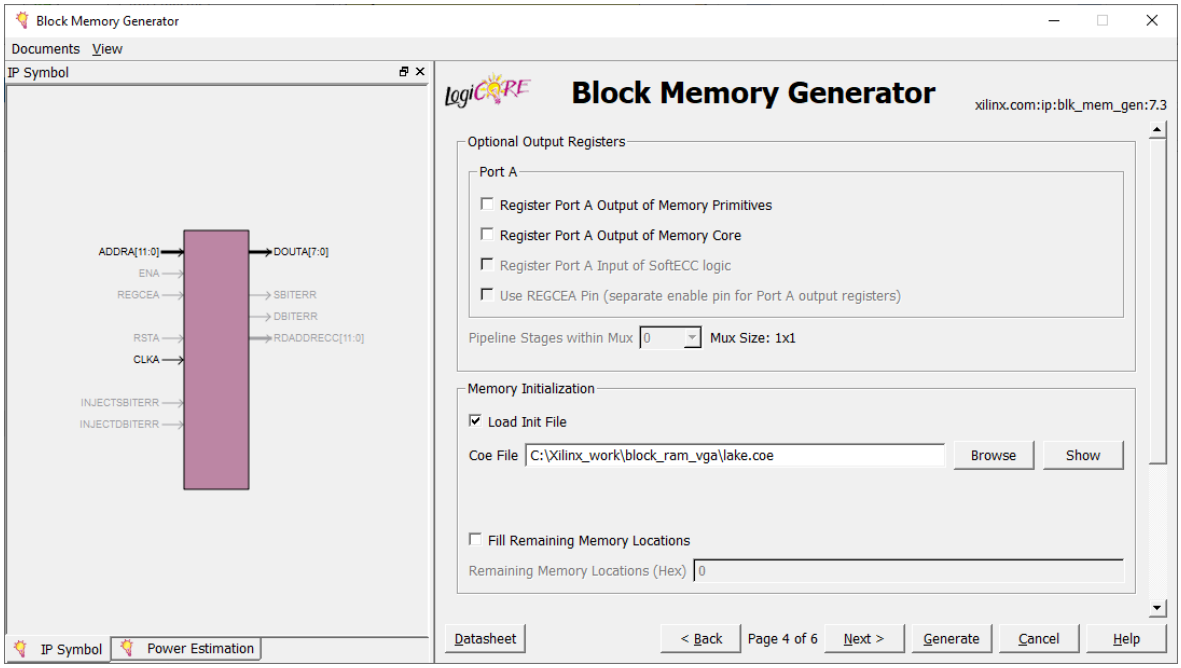


Po zakończeniu wyboru opcji pojawi się okno generatora pamięci, gdzie przechodzimy do 2 strony konfiguracji, na której wybieramy typ pamięci ustawiając opcję *Single Port ROM*.



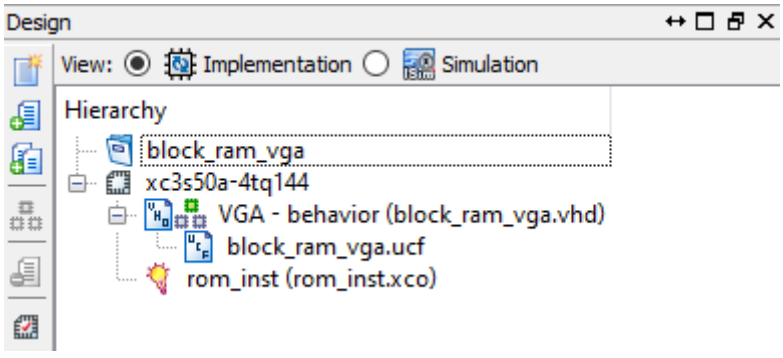
Na kolejnej, trzeciej stronie wpisujemy rozmiar wektora danych (*Memory Size -> Width*) oraz liczbę tych wektorów (*Memory Size ->Depth*). Jest to definiowana przez nas szerokość i głębokość pamięci. Nazwy komponentu nie zmieniamy. Zaznaczamy opcję *Always Enabled*.

Na 4 stronie konfiguratora pamięci pozostawiamy okno z domyślnymi ustawieniami poza pozycją *Memory Initialization,* gdzie ładujemy plik o nazwie *lake.coe* z wektorem inicjalizującym pamięć ROM, tworzoną z komórek pamięci BRAM.



Na kolejnych stronach konfiguracji pamięci pozostawiamy ustawienia domyślne i klikamy przycisk *Generate*.

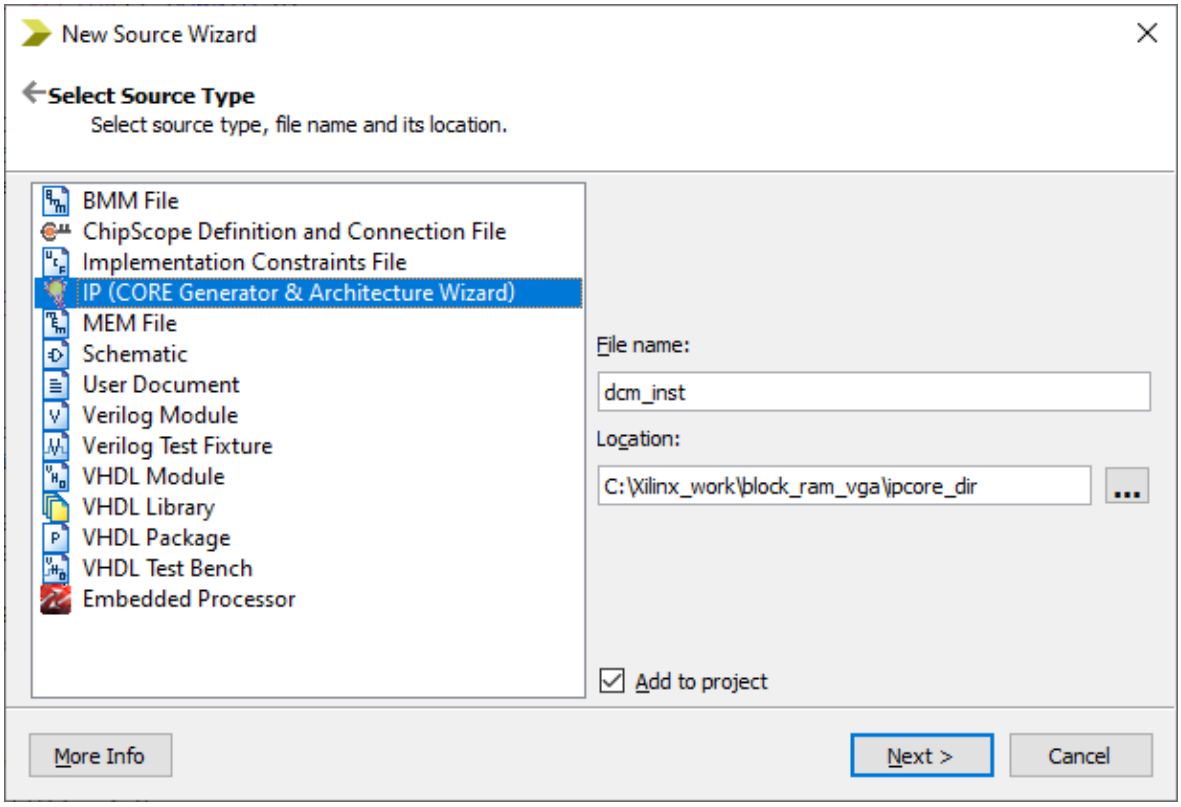
W zależności od wydajności komputera proces generacji pamięci ROM z komórek pamięci BRAM może zająć od kilkudziesięciu sekund do kilku minut. Po prawidłowym wykonaniu tej operacji w oknie z plikami źródłowymi projektu pojawi się dodatkowy plik o nazwie *rom_inst.xco*, który jest instancją pamięci ROM z wgranym już obrazkiem, który w dalszej części zajęć będzie wyświetlany na ekranie monitora VGA.

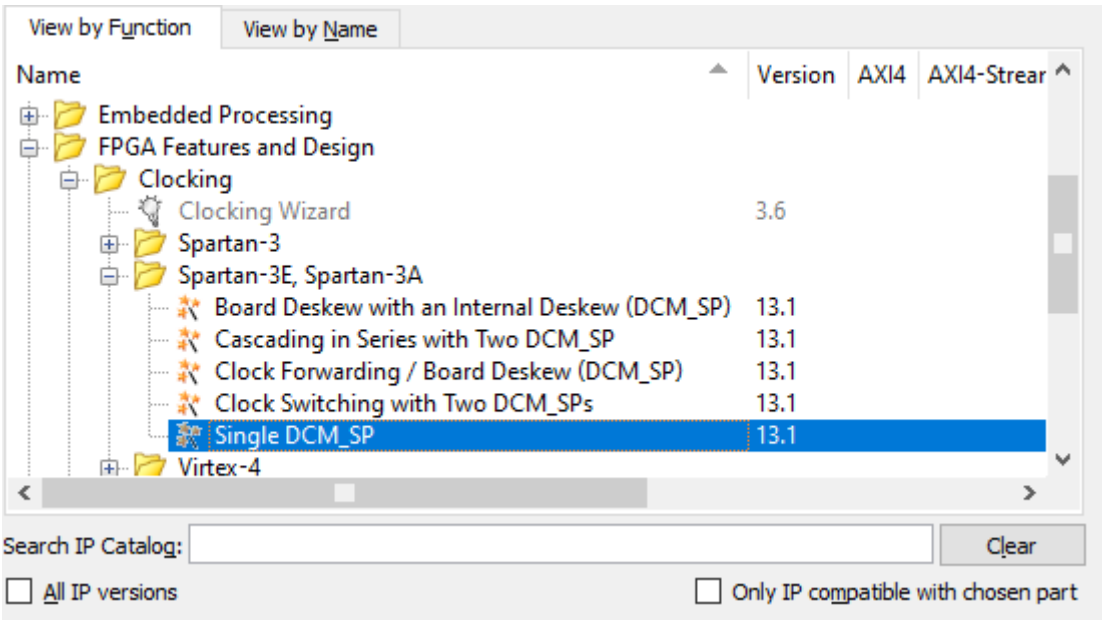


Po wygenerowaniu bloku pamięci można zobaczyć jego funkcjonalny opis w języku VHDL należy go zaznaczyć, w oknie *Processes* rozwinąć gałąź *CORE Generator* i dwukrotnie kliknąć na opcję *View HDL Functional Model.*

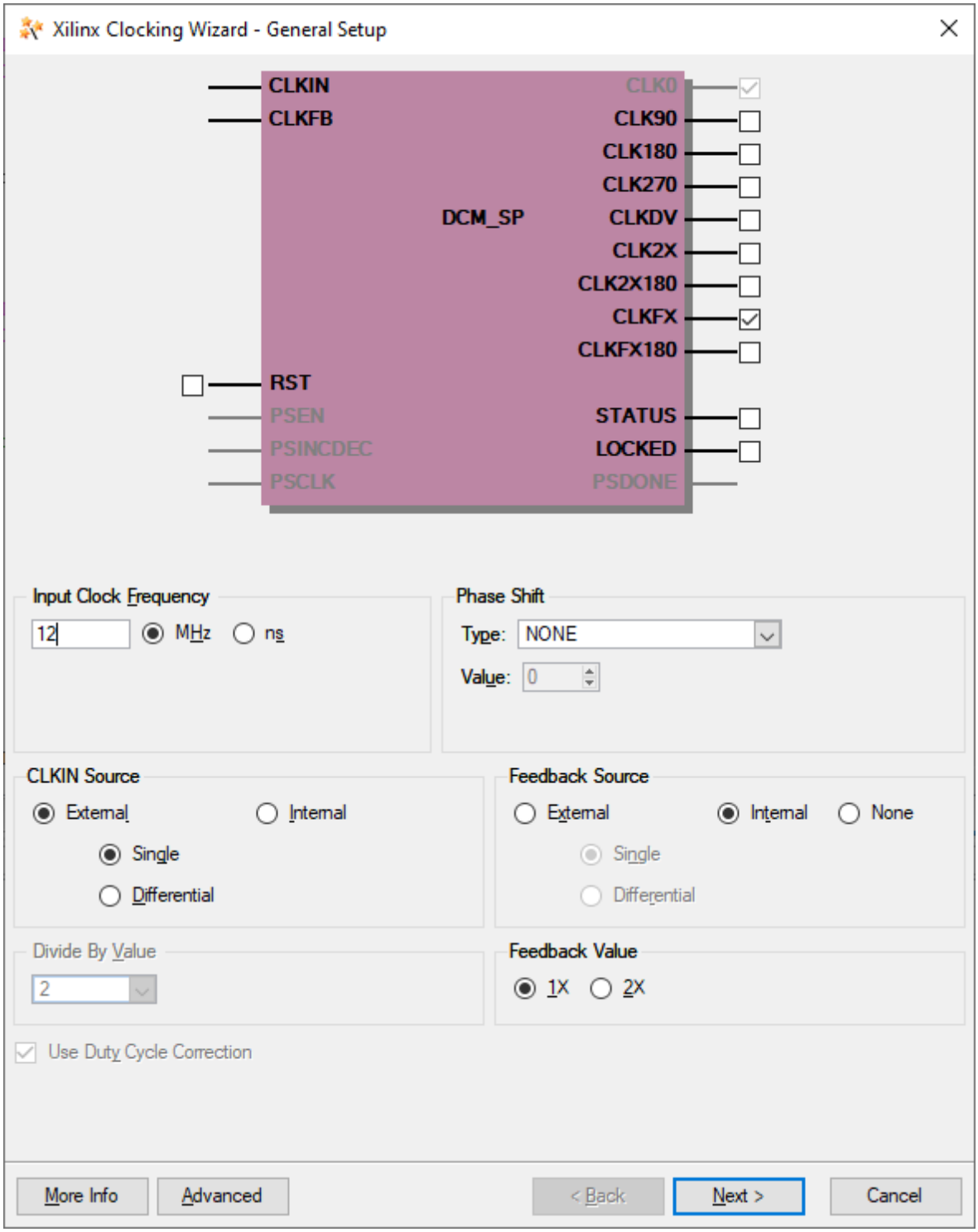
KROK 2: Dodanie do projektu cyfrowego menedżera sygnałów zegarowych (DCM), którego zadaniem będzie wygenerowanie sygnału zegarowego o częstotliwości 25 MHz.

Dodajemy nowe źródło do projektu i przy użyciu generatora *IP Core* dodajemy blok DCM wpisując *dcm_inst* jako jego nazwę.



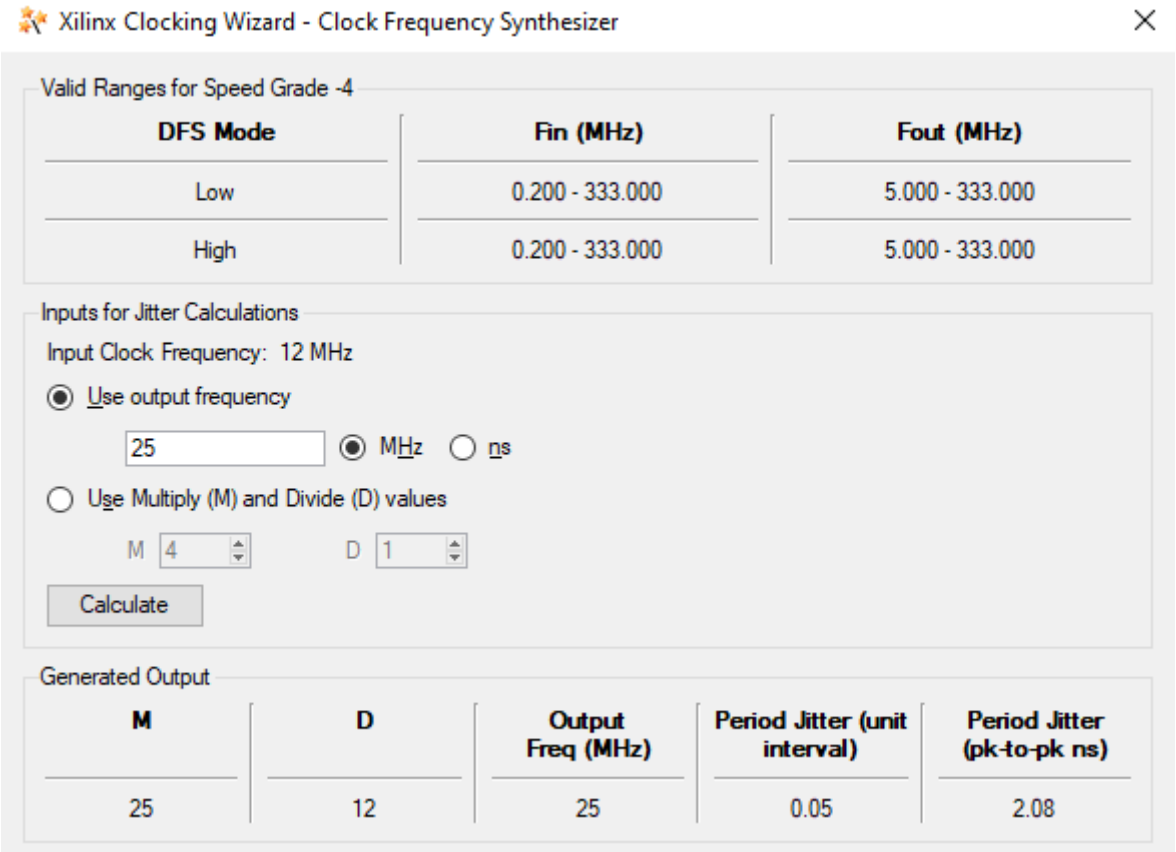


W kolejnym dwóch krokach potwierdzamy ustawienia i wybieramy język VHDL jako źródłowy dla instancji bloku DCM. Następnie ustalamy parametry sygnału zegarowego na wejściu – 12 MHz i zaznaczamy wyjście CLKFX, pozostałe odznaczamy.

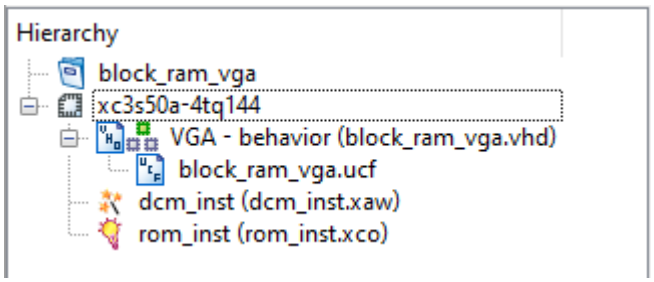


Po naciśnięciu przycisku *Next* w tym oknie i następnym konfigurator bloku DCM przeniesie nas na stronę syntezera częstotliwości zegara. W oknie syntezera podajemy wyjściową częstotliwość sygnału zegarowego równą 25 MHz. Jest ona niezbędna do prawidłowej pracy generatora sygnału wideo dla monitora z portem VGA, pracującego z

rozdzielczością 640 x 480 pikseli. Takie parametry pracy monitora VGA są zdefiniowane przez standard VESA.



W projekcie powinien się pojawić plik *dcm_inst.xaw*



UWAGA: Kod w języku VHDL z przykładem wykorzystania pamięci jest zawarty w plikach załączonych do ćwiczenia. W pliku *block_ram_vga.vhd* należy usunąć komentarze z istotnej części kodu:

```
component rom_inst
  port (
    clka : IN std_logic;
    addra : IN std_logic_vector(11 DOWNTO 0);
    douta : OUT std_logic_vector(7 DOWNTO 0));
  end component;
  ...

  ROM1 : rom_inst port map(clock,address,data);
```

```
oraz

component dcm_inst
port(
     CLKIN_IN : IN std_logic;
     CLKFX_OUT : OUT std_logic
    );
end component;
...

DCM1: dcm_inst PORT MAP(CLKIN_IN => clk , CLKFX_OUT => clk_25);
```

Po skompilowaniu projektu i wygenerowaniu pliku konfiguracyjnego *block_ram_vga.bit* należy zaprogramować układ docelowy i sprawdzić efekt działania systemu na ekranie monitora VGA.

Zadania:

1. Wygenerować inny wektor inicjalizujący dla pamięci ROM z obrazka przy użyciu skryptu *image_converter.m.* Pliki BMP o rozmiarach 64x64 piksele zostały udostępnione do ćwiczenia w osobnym katalogu.
2. Na ekranie obok obrazka oryginalnego wygenerować jego negatyw.
3. Przerobić projekt w ten sposób, aby wyświetlał obrazki monochromatyczne, w których piksele mają wartości binarne 0 lub 1. Redukcja szerokości magistrali danych do 1 bitu umożliwi wyświetlanie obrazków o rozdzielczości 128 x 128 pikseli.

References

- J. Majewski, P. Zbysiński – *Układy FPGA w przykładach.* Wydawnictwo BTC, Legionowo 2007.