

# ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

---

## Sprzętowa implementacja algorytmów

5. DCM - menedżer (syntezer) sygnałów zegarowych.

---

Lider projektu: Politechnika Warszawska

Autor: Łukasz Mik

Akademia Nauk Stosowanych w Tarnowie

# 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 Where other published and unpublished source materials have been used, these have been acknowledged.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

© Copyright 2021 - 2023 the [ENGINE](#) Consortium

Warsaw University of Technology (Poland)

International Hellenic University (IHU) (Greece)

European Lab for Educational Technology- EDUMOTIVA (Greece)

University of Padova (Italy)

University of Applied Sciences in Tarnow (Poland)

All rights reserved.



This document is licensed to the public under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

# 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.

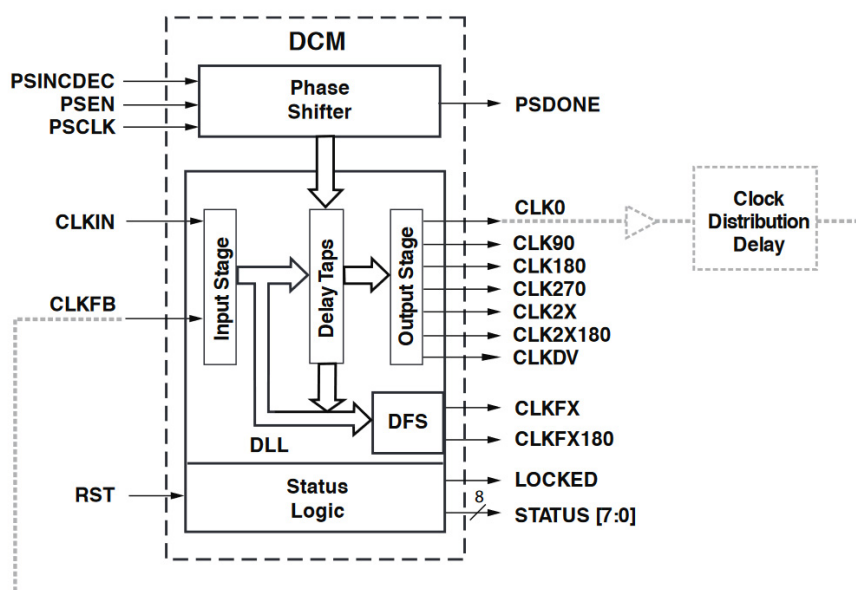
# I. Dystrybucja i zarządzanie sygnałami zegarowymi w rodzinie układów Spartan-3.

Z punktu widzenia projektanta system cyfrowego, wykorzystującego układy FPGA, jest odpowiednie taktowanie wszystkich elementów składowych tego systemu. Elementy logiczne rozmieszczone na powierzchni struktury półprzewodnikowej, ze względu na budowę wewnętrzną FPGA, nie są taktowane jednocześnie. Na czas dystrybucji sygnału mają wpływ następujące czynniki:

- długość trasy - odległość między źródłem a celem
- rodzaj trasy – zależny od zasobów połączeniowych, wykorzystanych do transportu sygnału
- liczba wejść taktowanych jednocześnie w danym segmencie ścieżki połączeniowej

Żeby zapobiec ryzyku zaniku efektu synchronizacji w układach FPGA stosuje się globalne linie zegarowe oznaczone jako GCLKx, gdzie x to numer linii. Przepisanie linii rozprowadzającej sygnał zegarowy do wyprowadzenia GCLK jest informacją dla programu projektowego, że użytkownik chce wykorzystać globalną linię zegarową.

Dodatkowo, aby zminimalizować wpływ niedoskonałości architektury FPGA na jakość implementowanych w nich projektów, firma Xilinx wyposaża układy Spartan 3A w bloki DCM (ich liczba zależy od zasobów logicznych FPGA).



Schemat blokowy DCM w układach Spartan 3

DCM składa się z 4 podstawowych elementów: syntezerę częstotliwości DFS (*Digital Frequency Synthesizer*), pętli DLL (*Delay Locked Loop*), programowalnego przesuwnika fazy (*Phase Shifter*) oraz zespołu logiki *Status Logic*.

Przy użyciu DCM jest możliwe kompensowanie różnicy faz sygnałów zegarowych w fizycznie różnych miejscach krzemowej struktury układu. Dzięki wbudowanej pętli DLL z regulowaną linią opóźniającą jest możliwe syntezywanie wewnętrznych sygnałów zegarowych (w tym mnożenie lub dzielenie częstotliwości sygnału zegarowego podłączonego z zewnątrz). Sygnał wejściowy dla DLL jest podawany na wejście CLKIN. Wejście CLKFB służy do opcjonalnego podania sygnału zwrotnego dla pętli sprzężenia zwrotnego, dzięki czemu blok DLL może monitorować jakość generowanego sygnału taktującego. Blok DLL wyposażono w cztery wyjścia sygnałów będących kopiami sygnału z wejścia CLKIN, z czego 3 są przesunięte w fazie względem niego o: 90°, 180° i 270°. Na tych wyjściach jest możliwe uzyskanie sygnałów o wypełnieniu 50% pod warunkiem ustawienia parametru *DUTY\_CYCLE\_CORRECTION* na *TRUE*. Na wyjściach CLK2X i CLK2X180 jest generowany sygnał o wypełnieniu 50%, częstotliwości dwukrotnie wyższej niż częstotliwość sygnału wejściowego i fazach 0° i 180°. Na wyjściu CLKDV można uzyskać sygnał o częstotliwości podzielonej przez następujące wartości: 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15 lub 16. Współczynnik wypełnienia sygnału na tym wyjściu wynosi 50%, gdy współczynnik podziału jest liczbą całkowitą.

Sygnały na wyjściach CLKFX i CLKFX180 (przesunięte względem siebie o 180°) są generowane za pomocą syntezerę częstotliwości DFS, na podstawie sygnału podawanego na wejście CLKIN. Zakres dopuszczalnych częstotliwości wejściowych syntezerę zawiera się w przedziale od 1 do 280 MHz. Częstotliwość sygnału na wspomnianych wyjściach otrzymuje się przez podzielenie lub pomnożenie częstotliwości sygnału wejściowego zgodnie ze wzorem:

$$f_{\text{CLKFX}} = f_{\text{CLKIN}} \times M/D$$

gdzie:  $M = \{2 \dots 32\}$ ,  $D = \{1 \dots 32\}$

Blok DCM jest wyposażony również w wejście zerujące RST, które przywraca konfigurację domyślną (zdefiniowaną przez użytkownika) oraz wyjście LOCKED, służące do sygnalizacji zsynchronizowania się DLL z sygnałem na wejściu CLKIN.

W zaawansowanych projektach (przy wysokich częstotliwościach taktowania) konieczne jest zastosowanie przesuwnika fazy, który może być sterowany dynamicznie. Na etapie tego kursu nie będzie on jednak wykorzystywany.

## II. Zwiększanie częstotliwości sygnału zegarowego za pomocą bloku DCM.

Tworzymy nowy projekt o nazwie *dcm\_test1* w ISE WebPack wprowadzając odpowiednie parametry docelowego układu FPGA. Tworzymy nowy plik źródłowy VHDL (VHDL Module) o nazwie *higher\_freq*. W jednostce projektowej dodajemy jeden port wejściowy o nazwie CLK i jeden wyjściowy o nazwie LED. Po dodaniu portów deklaracja jednostki projektowej wraz z częścią nagłówkową będzie wyglądała następująco:

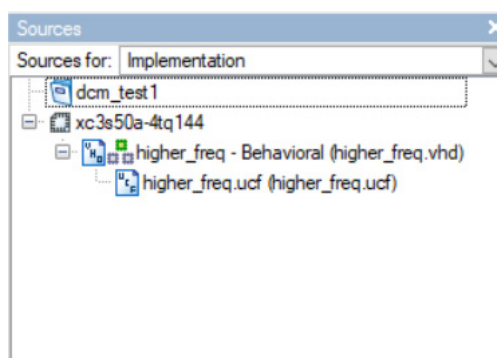
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity higher_freq is
    port ( CLK : in  std_logic;
          LED : out std_logic);
end higher_freq;
```

W kolejnym kroku dodajemy plik UCF z przypisaniem portów jednostki do pinów FPGA. Przy tworzeniu nowego pliku źródłowego należy wybrać *Implementation Constraints File* i jako nazwę pliku wpisać *higher\_freq*. W utworzonym pliku należy dodać 2 linie z przypisaniem pinów.

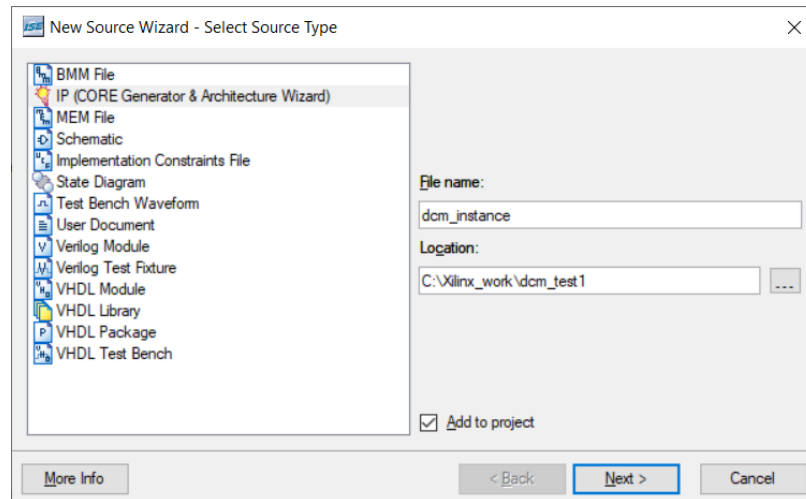
```
NET "CLK" LOC = P129 | IOSTANDARD = LVCMOS33 | PERIOD = 12MHz;
NET "LED" LOC = P46 | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
```

Po prawidłowym dodaniu plików okno z podglądem plików źródłowych powinno wyglądać następująco:

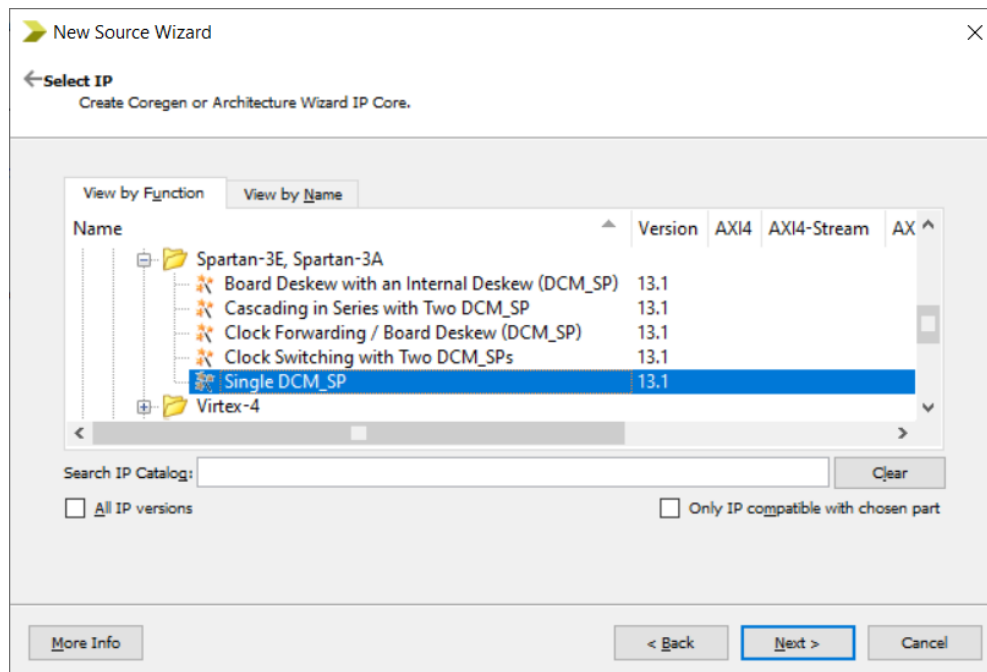


Jak można zauważyć nazwa projektu nie musi się pokrywać z nazwą plików źródłowych. Nazwy plików źródłowych również nie muszą się pokrywać z nazwą jednostki projektowej. Taki styl nazewnictwa plików ułatwia tylko pracę z projektami zawierającymi wiele komponentów.

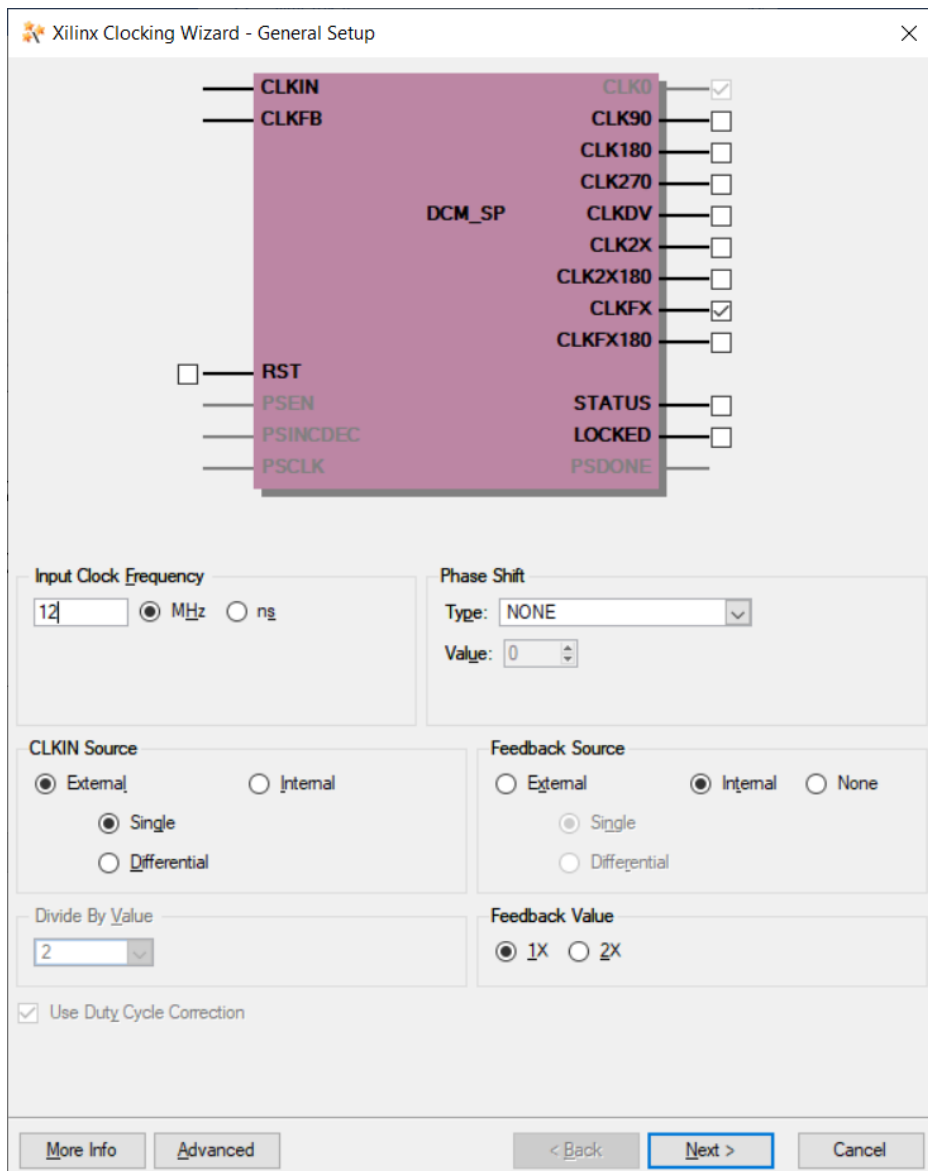
W kolejnym kroku dodajemy do projektu gotowy blok IP Core przy użyciu narzędzia *IP (CORE Generator & Architecture Wizard)*. Klikając prawym klawiszem w oknie *Sources* lub wybierając z menu poziomego opcję *Project* → *New Source ...* Następnie wybieramy opcję *IP (CORE Generator & Architecture Wizard)* i wpisujemy *dcm\_instance* jako nazwę pliku.



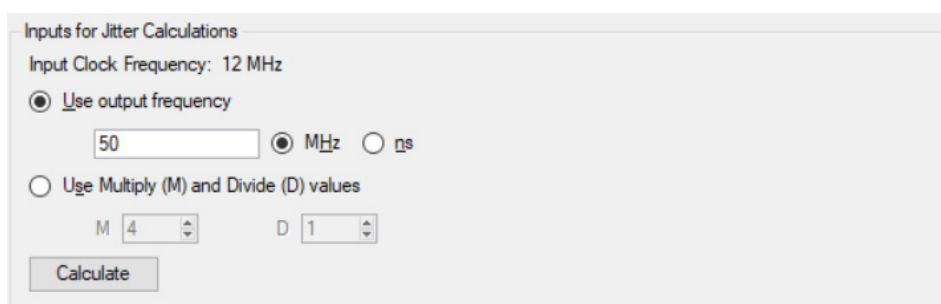
Po kliknięciu przycisku *Next* i przejściu do następnego okna rozwijamy grupę *FPGA Features and Design* → *Clocking* → *Spartan 3E, Spartan 3A* i wybieramy opcję *Single DCM\_SP*.



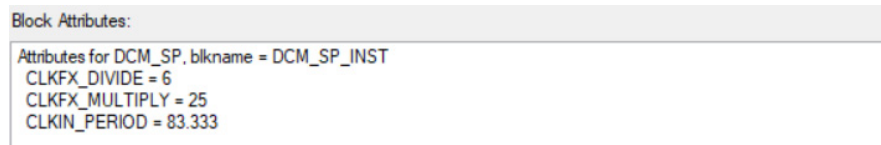
Po chwili otworzy się okno z wyborem typu pliku wyjściowego – należy wybrać VHDL i kliknąć *OK*, po czym otworzy się okno z ustawieniami bloku DCM.



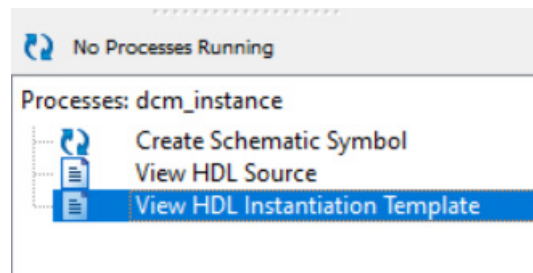
W tym oknie wpisujemy częstotliwość wejściowego sygnału zegarowego: 12 MHz. Oznaczamy linie RST i LOCKED (nie będziemy z nich korzystać) oraz zaznaczamy linię CLKFX – wyjście zegara o wyższej częstotliwości. Po zatwierdzeniu ustawień przyciskiem *Next*, zatwierdzamy w kolejnym oknie użycie globalnych buforów dla wskazanych w bloku DCM linii zegarowych. W kolejnym oknie pojawi się konfiguracja syntezy częstotliwości (DFS). Ustawiamy docelową częstotliwość wyjściową na poziomie 50 MHz.



Po przejściu dalej pojawi się okno z podsumowaniem ustawień DFS.



Współczynnik dzielenia częstotliwości sygnału zegarowego z wejścia CLKIN został ustawiony na 6 natomiast współczynnik mnożenia na 25. Częstotliwość na wyjściu CLKFX będzie w takim przypadku wynosić:  $f_{CLKFX} = 12 \text{ MHz} \times (25/6) = 50 \text{ MHz}$ , czyli tyle ile podaliśmy w oknie konfiguracyjnym. Po prawidłowym wygenerowaniu bloku DCM do projektu zostanie dodany plik o nazwie *dcm\_instance.xaw*. Przez podwójne kliknięcie na nazwę tego pliku ponownie wejdziemy do trybu konfiguracji DCM. Jeśli zaznaczymy nazwę pliku to w oknie *Processes* możemy uruchomić podgląd szablonu komponentu, który trzeba wstawić do pliku z opisem architektury.



W oknie edycyjnym otworzy się zawartość pliku *dcm\_instance.vhi* z której kopiujemy ten fragment:

```
COMPONENT dcm_instance
PORT(
    CLKIN_IN : IN std_logic;
    CLKFX_OUT : OUT std_logic;
    CLKIN_IBUFG_OUT : OUT std_logic;
    CLK0_OUT : OUT std_logic
);
END COMPONENT;
```

Następnie wklejamy go do wnętrza architektury w części deklaracyjnej tj. przed słowem kluczowym *begin* rozpoczynającym opis behawioralny.

Do przetestowania poprawności działania projektu zdefiniujemy wewnątrz architektury sygnał o nazwie *clk\_50MHz*, który podłączymy do wyjścia CLKFX\_OUT bloku DCM. Następnie wstawimy proces, który na liście wrażliwości będzie miał sygnał *clk\_50MHz*, a zadaniem tego procesu będzie generowanie sygnału *clk\_1Hz* o częstotliwości 1 Hz, który bezpośrednio podłączymy do portu wyjściowego LED.



Na końcu będzie potrzebne jeszcze podłączenie bloku DCM do sygnałów wewnątrz architektury. Połączenie wstawionego komponentu odbywa się przez słowa kluczowe **port map**, po których w nawiasie łączy się sygnały w architekturze do portów tego komponentu. Ponieważ będziemy korzystać tylko z linii CLKIN\_IN oraz CLKFX\_OUT pozostałe możemy zostawić niepodłączone jak pokazano na poniższym przykładzie.

```
DCM1 : dcm_instance port map(CLKIN_IN => CLK , CLKFX_OUT => clk_50MHz);
```

Etykieta komponentu np. DCM1 jest niezbędna podczas umieszczania go w projekcie. Kompletny opis architektury przyjmie postać jak na poniższym listingu.

```
architecture Behavioral of higher_freq is
```

```
COMPONENT dcm_instance
```

```
PORT(
```

```
    CLKIN_IN : IN std_logic;
```

```
    CLKFX_OUT : OUT std_logic;
```

```
    CLKIN_IBUFG_OUT : OUT std_logic;
```

```
    CLK0_OUT : OUT std_logic
```

```
);
```

```
END COMPONENT;
```

```
signal clk_50MHz : std_logic := '0';
```

```
signal clk_1Hz : std_logic := '0';
```

```
begin
```

```
process(clk_50MHz)
```

```
variable counter_1 : integer := 0;
```

```
begin
```

```
    if rising_edge(clk_50MHz) then
```

```
        if counter_1 < 25000000 then
```

```
            counter_1 := counter_1 + 1;
```

```
        else
```

```
            counter_1 := 0;
```

```
            clk_1Hz <= not clk_1Hz;
```

```
        end if;
```

```
    end if;
```

```
end process;
```

```
LED <= clk_1Hz;
```

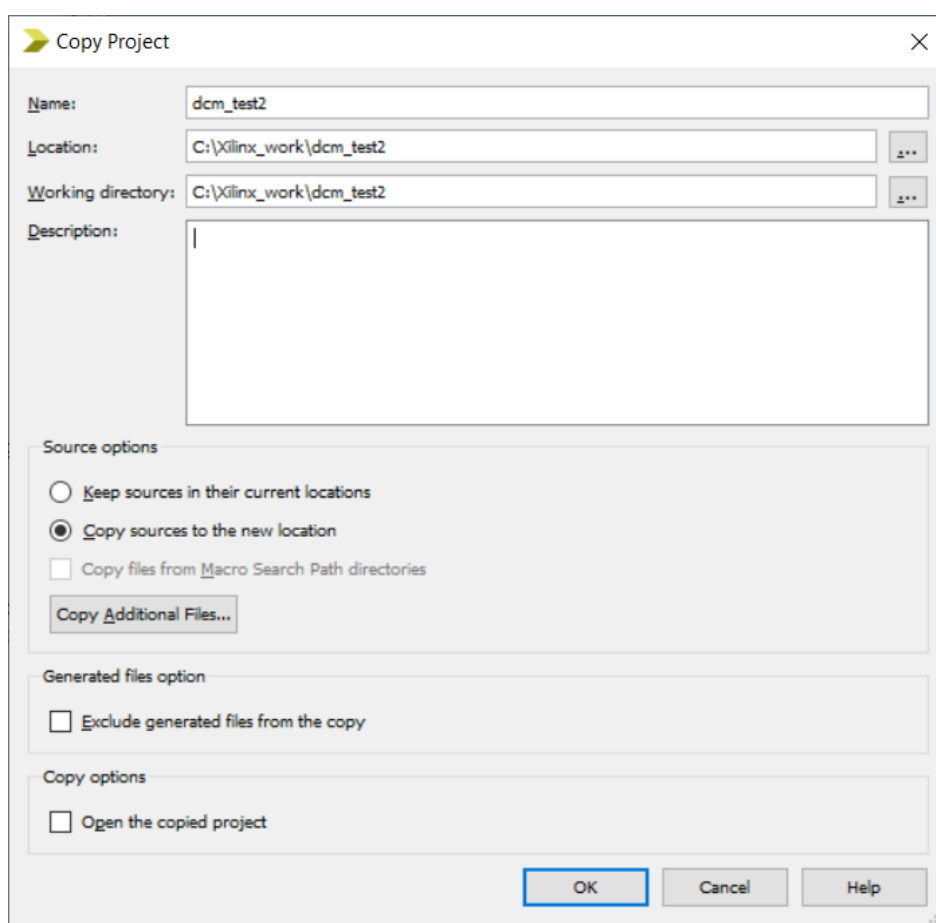
```
DCM1 : dcm_instance port map(CLKIN_IN => CLK , CLKFX_OUT => clk_50MHz);
```

```
end Behavioral;
```

Po skompilowaniu projektu i wygenerowaniu pliku konfiguracyjnego należy zaprogramować FPGA i sprawdzić czy dioda LED miga z częstotliwością 1 Hz. Należy pamiętać, że plik konfiguracyjny będzie miał nazwę *higher\_freq.bit* (lub *higher\_freq.bin* jeśli użytkownik ustawi to w opcjach generatora pliku).

### III. Zmniejszenie częstotliwości sygnału zegarowego za pomocą bloku DCM.

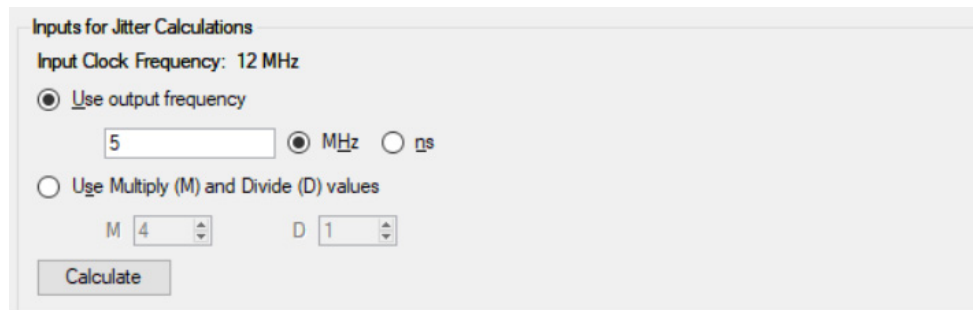
Przykład generowania sygnału zegarowego o częstotliwości niższej niż wejściowy sygnał zegarowy omówimy na przykładzie z poprzedniego punktu. W tym celu należy skopiować projekt pod inną nazwą. Z menu *File* wybieramy opcję *Copy Project ...* i w oknie dialogowym, które się pojawi należy podać nową nazwę projektu tj. *dcm\_test2*.



Następnie zamykamy bieżący projekt (*File* → *Close Project*) i otworzyć projekt *dcm\_test2*. Ponieważ pliki źródłowe *\*.vhd* i *\*.ucf* wciąż mają starą nazwę należy je usunąć z projektu. Następnie zmieniamy ich nazwy w folderze projektu na *lower\_freq.vhd* i *lower\_freq.ucf* i ponownie dodajemy do projektu z nowymi nazwami. Zmieniamy też nazwę jednostki projektowej na *lower\_freq*.

```
entity lower_freq is
    port ( CLK : in  std_logic;
          LED : out std_logic);
end lower_freq;
```

Następnie klikamy dwukrotnie na plik o nazwie *dcm\_instance.xaw* dzięki czemu przejdziemy do okna dialogowego z konfiguracją parametrów bloku DCM. Jedyne co zmieniamy w konfiguracji to częstotliwość sygnału zegarowego na wyjściu CLKFX\_OUT. Ustawiamy wartość 5 MHz.



Inputs for Jitter Calculations  
Input Clock Frequency: 12 MHz  
 Use output frequency  
5 MHz  ns  
 Use Multiply (M) and Divide (D) values  
M 4 D 1  
Calculate

Konfigurator sam obliczy parametry CLKFX\_DIVIDE oraz CLKFX\_MULTIPLY dla syntezy częstotliwości.

```
Block Attributes:  
Attributes for DCM_SP, blkname = DCM_SP_INST  
CLKFX_DIVIDE = 12  
CLKFX_MULTIPLY = 5  
CLKIN_PERIOD = 83.333
```

W starym opisie architektury zmieniamy nazwę sygnału clk\_50MHz na clk\_5MHz. W procesie generującym sygnał clk\_1Hz zmieniamy wartość graniczną licznika z 25000000 na 2500000.

## ZADANIA:

- Wykorzystaj wyjścia CLKFX i CLKFX180, do sterowania 2 różnymi diodami LED. Pamiętaj o tym, że sterowanie tymi diodami musi odbywać się 2 różnymi sygnałami taktującymi 1 Hz powiązanych z wymienionymi wyjściami bloku DCM.
- Wykorzystaj wyjścia CLK0, CLK90, CLK180 i CLK270 do sterowania 4 różnych diod LED. Pamiętaj o zależności sygnałów taktujących 1 Hz wspomnianej w poprzednim punkcie.

## References

- J. Majewski, P. Zbysiński – *Układy FPGA w przykładach*. Wydawnictwo BTC, Legionowo 2007.
- UG331 – Spartan-3 Generation FPGA User Guide  
<https://docs.xilinx.com/v/u/en-US/ds529>