Custom Computing & Embedded Systems

Xilinx ISE WebPack – szybki start

1. Cel ćwiczenia

Celem ćwiczenia jest szybkie zapoznanie się z edytorem schematów i symulatorem wbudowanym w pakiet ISE firmy Xilinx.

2. Omówienie zadania projektowego F0/4

Korzystając ze skryptu *zadania_EC.tcl* wygenerowane zostały zadania projektowe dla danych: *12345 Adam Adamski*. Funkcja F0/4 z powyższego zestawu zadań przedstawia się następująco:

```
F0/4(A,B,C,D) = ∑{2 3 4 5 10 12 14}
---------------
| DEC |  BIN  |
|-----|-------|
|  2  |  0010 |
|  3  |  0011 |
|  4  |  0100 |
|  5  |  0101 |
| 10  |  1010 |
| 12  |  1100 |
| 14  |  1110 |
---------------
```

Zgodnie z powyższą tablicą prawdy możemy zapisać tablicę Karnaugh:

A B \ C D	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	0	0	1
10	0	0	0	1

W wyniku minimalizacji otrzymujemy funkcję:

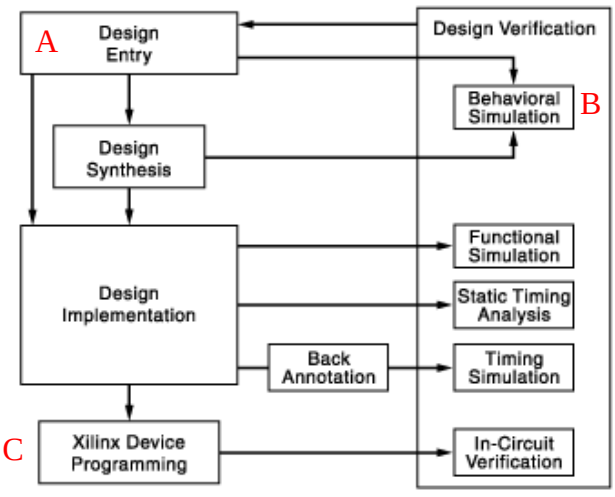
```
F0/4(A,B,C,D) = |B*C*|D + |A*|B*C + A*B*|D + |A*B*|C
```

gdzie | oznacza negację zmiennej, a * iloczyn logiczny.
W dalszej części ćwiczenia funkcja F0/4 będzie realizowana w układzie FPGA i badana analizatorem stanów logicznych.

3. Cykl projektowania

W celu implementacji projektu należy skorzystać z oprogramowania ISE firmy Xilinx. Cykl projektowania z wykorzystaniem układów rekonfigurowalnych przedstawiony jest na schemacie obok. W czasie wykonywania ćwiczeń w laboratorium elektroniki cyfrowej wykorzystywane będą tylko niektóre programy z pakietu. W instrukcji skupimy się na etapie wprowadzania projektu (A), symulacji behawioralnej (B) oraz programowaniu układu FPGA – etap C.

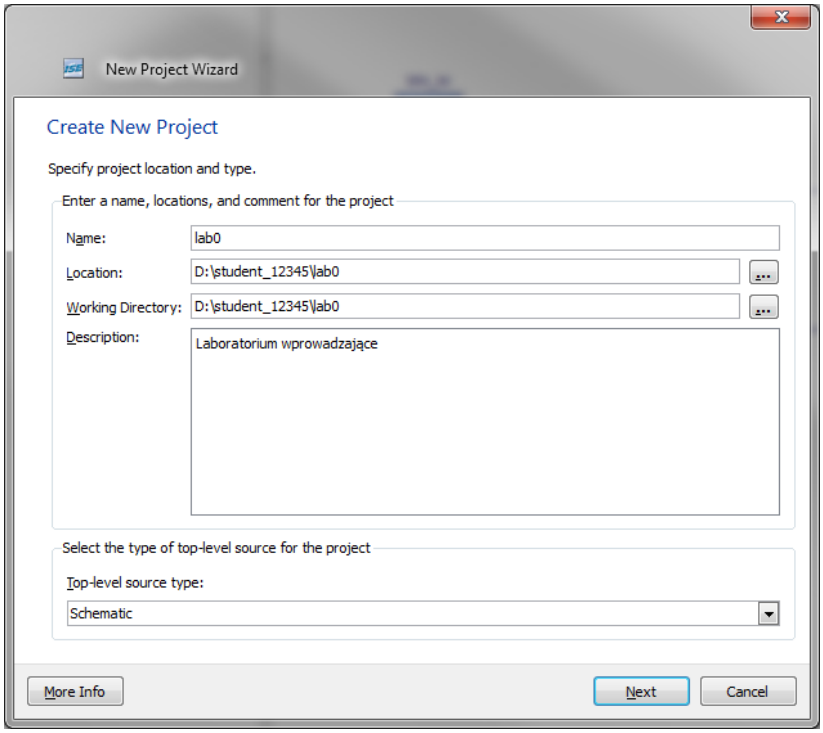
etap A – edytor schematów ECS
etap B – symulator (ISim lub Modelsim)
etap C – programator Adept lub iMPACT



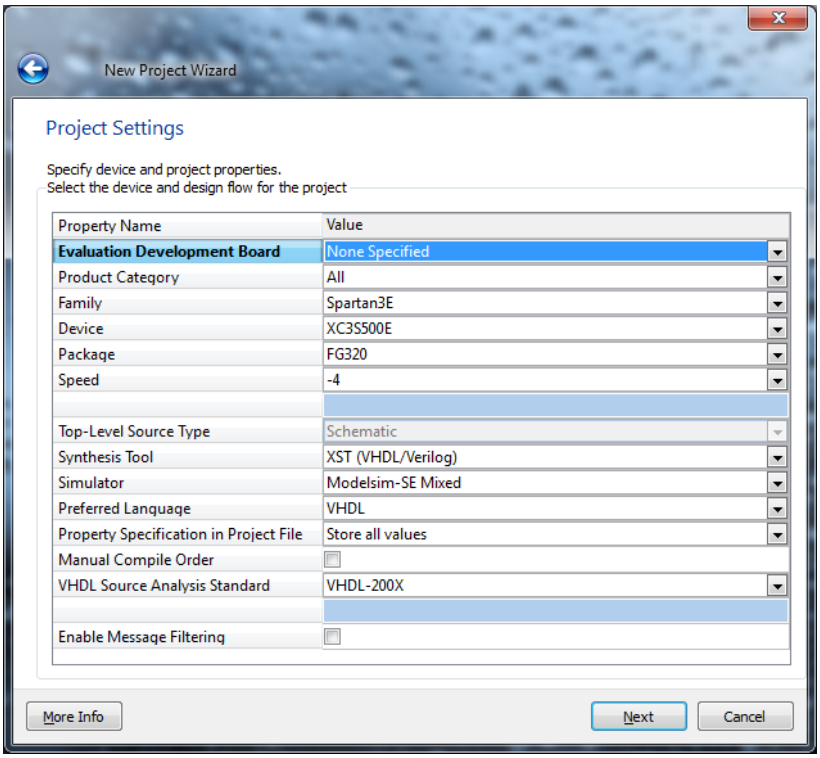
4. Utworzenie projektu

- uruchom program *Project Navigator* z grupy ISE Design Tools
- wybierz polecenie *New Project...* z menu *File*
- ustaw opcję Top-level source type: *Schematic*

◦ *projekty należy tworzyć na dysku D: z nazwą <nrlab_nrindeksu>*



- zdefiniuj parametry projektu zgodnie z poniższym oknem dialogowym:

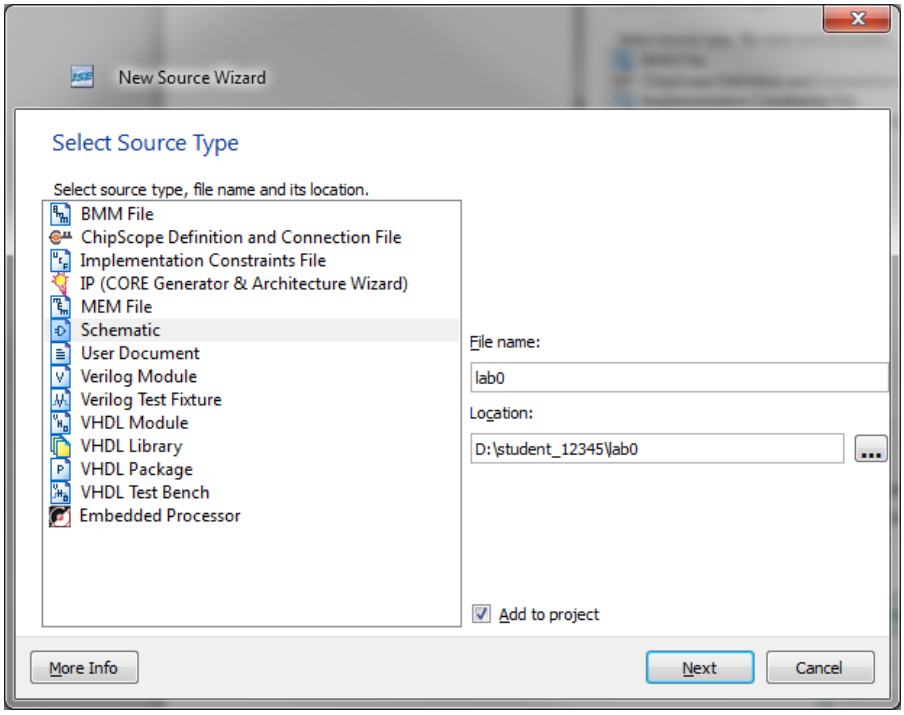


Uwaga: wybór symulatora (ISim, Modelsim-SE, Modelsim-PE) uzgodnić z prowadzącym zajęcia.

4.1. Utworzenie nowego schematu

Nowe źródło projektowe tworzymy używając polecenia *New Source...* z menu *Project* i wybierając odpowiedni typ źródła. W tym tutorialu będziemy realizowali opis strukturalny układu cyfrowego, dlatego należy utworzyć nowy schemat w katalogu głównym projektu.

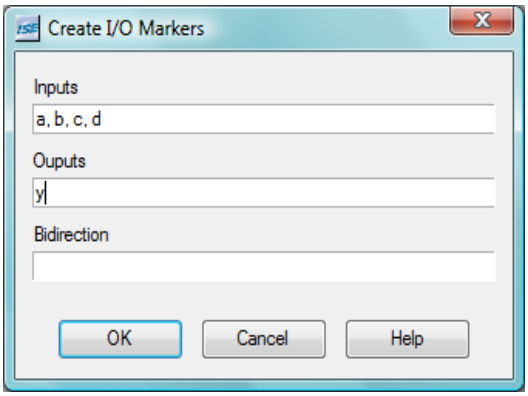
- wybierz polecenie **New Source**... z menu **Project**
- wybierz typ źródła 'Schematic', nadaj nazwę bez użycia spacji i liter diakrytyzowanych.



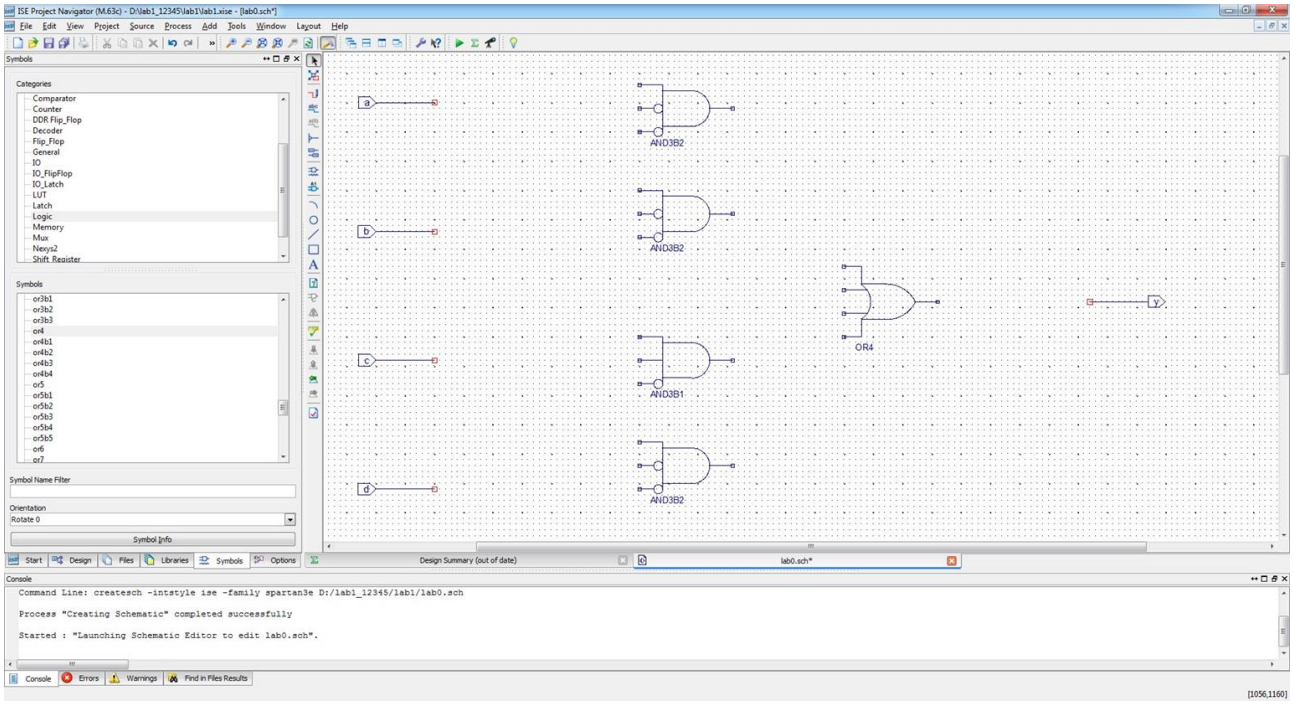
4.2. Realizacja zadania F0/4

Po utworzeniu nowego schematu zostaje on automatycznie otwarty w oknie edycji (prawa strona interfejsu użytkownika). Należy teraz dodać do schematu znaczniki interfejsów wejścia/wyjścia (*I/O Markers*) aby umożliwić komunikację projektowanego układu ze światem zewnętrznym. Kolejnym krokiem jest wykonanie schematu złożonego z podstawowych bramek i kontrola poprawności wykonania schematu.

- przy otwartym oknie schematu **lab0** wybierz polecenie *Tools -> Create I/O Markers* (uwaga: polecenie to działa tylko dla schematów, które nie były jeszcze edytowane); wypełnij okno dialogowe nazwami potrzebnych wyprowadzeń wg rysunku:



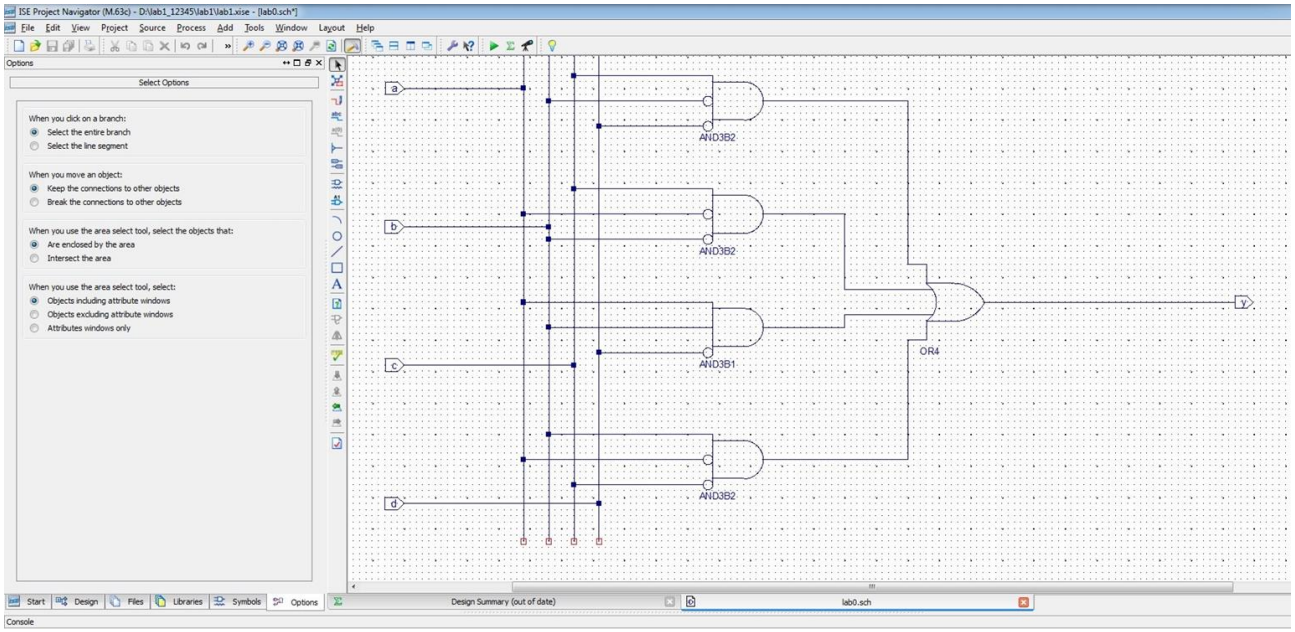
- w zakładce *Symbols* wybierz kategorię *Logic* i zapoznaj się z dostępnymi elementami typu AND i OR; posłużą one do budowy schematu realizującego równanie boolowskie funkcji F0/4.



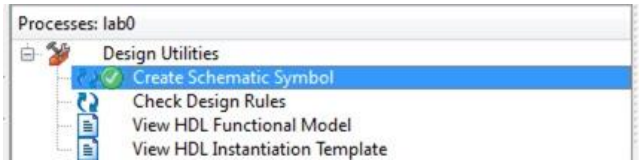
- wybierz odpowiednie elementy do realizacji funkcji F0/4 (można wykorzystać inwertery lub bramki z wejściami zanegowanymi z oznaczeniem Bx, gdzie x jest liczba wejść zanegowanych)

```
F0/4(A,B,C,D) = |B*C*|D + |A*|B*C + A*B*|D + |A*B*|C
```

- połącz elementy przy pomocy polecenia *Add -> Wire*

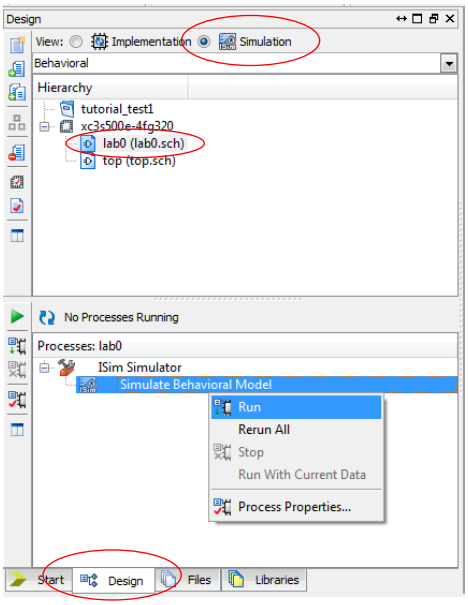


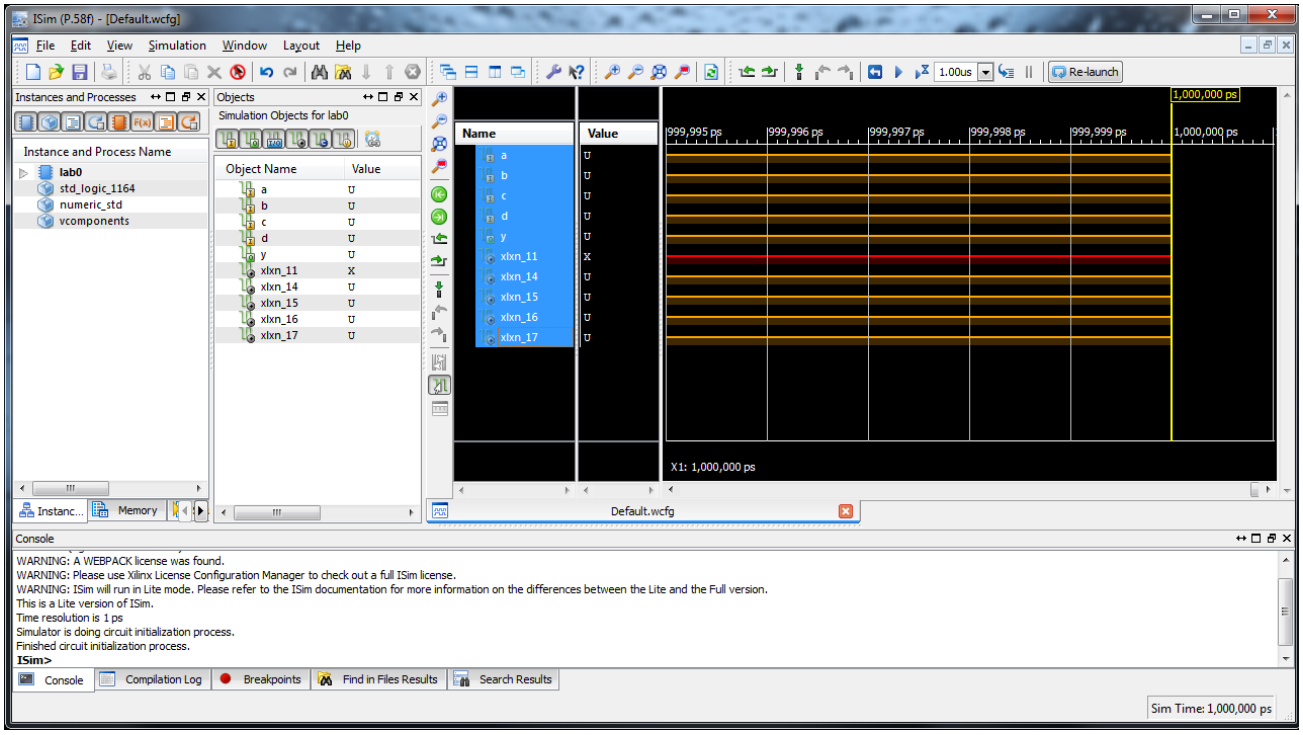
- zapisz zmiany dokonane na schemacie
- z menu *Tools* wybierz polecenie *Check Schematic* w celu sprawdzenia poprawności wykonania połączeń; sprawdź efekty wykonania polecenia w oknie transkrypcji (*Console*)
- zaznacz *lab0.sch* w oknie źródeł (zakładka *Design*, okno *Hierarchy*); w oknie *Processes* rozwiń grupę *Design Utilities*
- w oknie *Processes -> Design Utilities* wybierz polecenie *Create Schematic Symbol* i utwórz symbol dla elementu **lab0** (tak utworzony symbol może być używany na innych schematach w celu budowania bardziej złożonych, wielopoziomowych projektów)



5. Symulacja behawioralna (ISim)

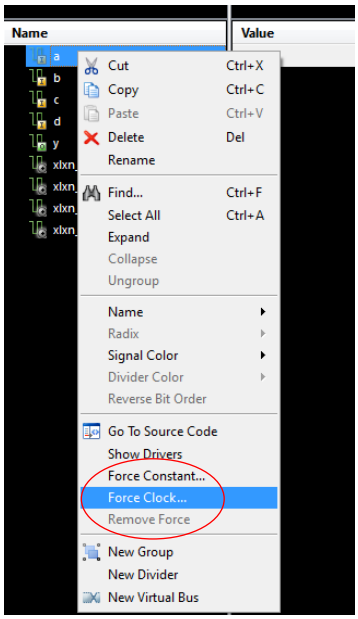
- w zakładce *Design* ustawiamy rodzaj widoku na *Simulation*, wybieramy plik *lab0.sch* i uruchamiamy symulator poleceniem *Simulate Behavioral Model -> Run.* Symulator uruchomiony zostanie jako program w nowym oknie z automatycznie otwartym projektem odpowiadającym symulowanemu plikowi *lab0.sch*



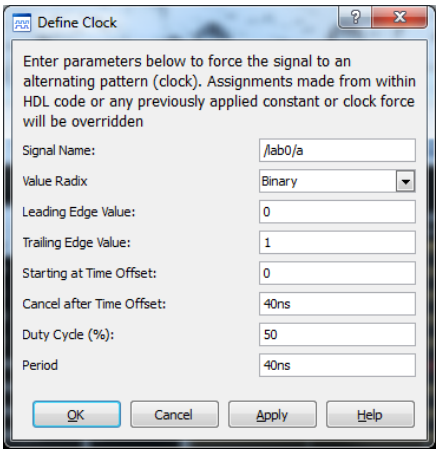


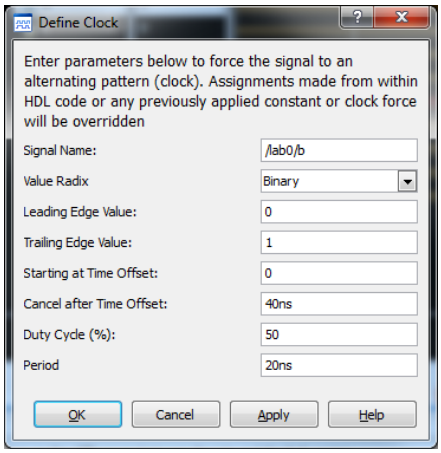
Uwaga: wyniki symulacji w powyższym oknie są nieokreślone (U), ponieważ nie zostały zdefiniowane wymuszenia; aby otrzymać poprawne wyniki należy wrócić do czasu 0 symulatora (polecenie *restart*), zdefiniować wymuszenia i uruchomić symulację na określony czas (np. poleceniem *run 80ns*); sposoby definiowania wymuszeń opisano w następnym punkcie.

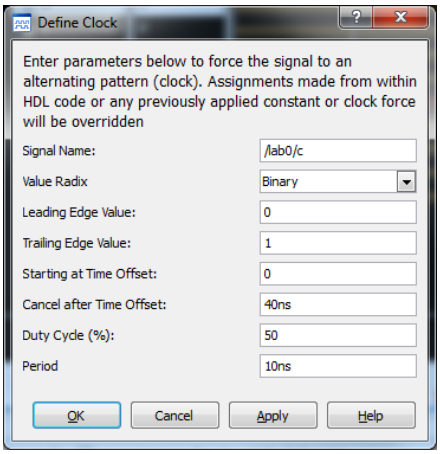
- w oknie symulatora istnieje możliwość realizacji wymuszeń, poprzez zaznaczenie danego sygnału i wybór jednego z poleceń:
  - *Force Constant…* – wymuszenie stałe o określonej długości,
  - *Remove Force* – usunięcie wymuszenia,
  - *Force Clock…* – wymuszenie okresowe.

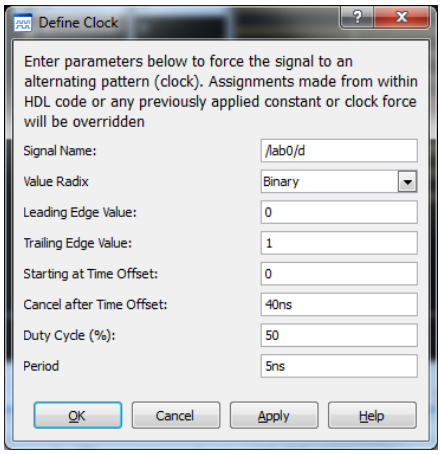


- zrealizuj wymuszenia dla przetestowania elementu *lab0* za pomocą polecenia *Force Clock…* zgodnie z poniższymi rysunkami





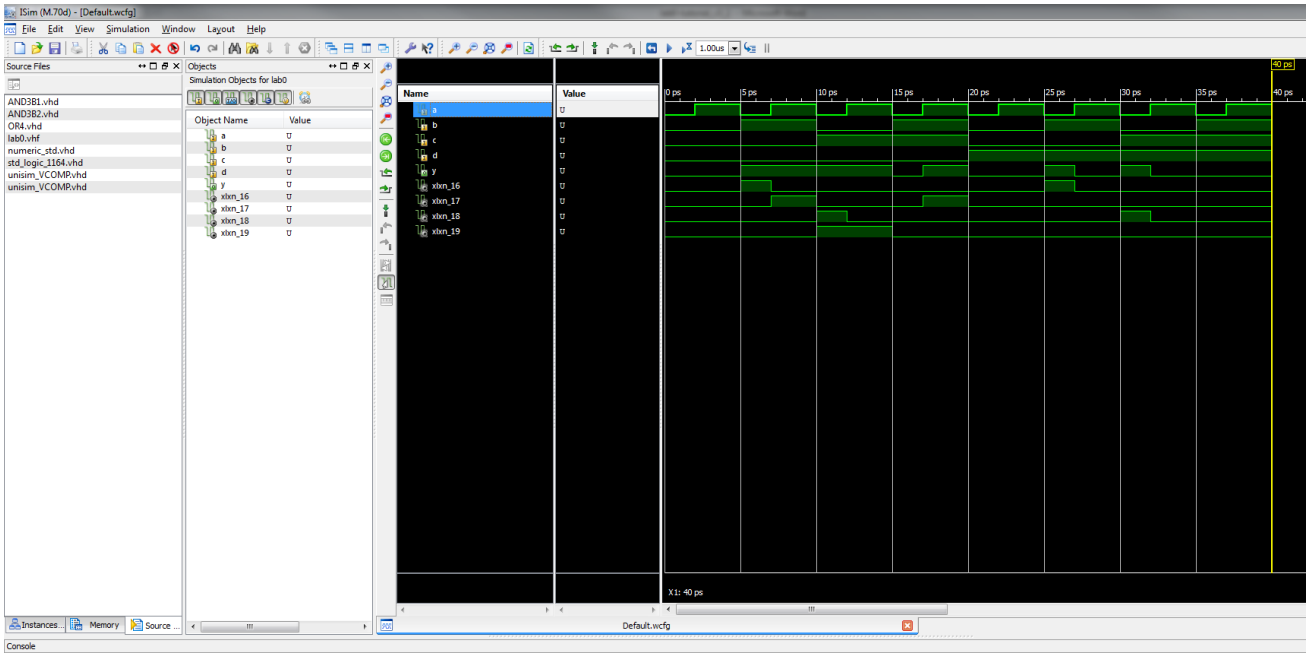




Uwaga: otrzymamy w ten sposób fragmenty przebiegów okresowych tworzących tablicę prawdy dla funkcji 4 zmiennych; każdy z kolejnych przebiegów jest dwa razy szybciej zmiennych od poprzedniego; najszybciej zmienny jest sygnał /lab0/d – jest to nasz najmłodszy bit; wszystkie wymuszenia zostają zawieszone po 40ns (*Cancel Time*); domyślną jednostką tego symulatora jest 1ps, aby definiować czasy innego rządu należy podać jednostkę czasu (np. *ns, us, ms, sec*); identyczne wymuszenia można osiągnąć poprzez komendy wydane w konsoli:

```
isim force add {/lab0/d} 0 -radix bin -value 1 -radix bin -time 2.5 ns -repeat 5 ns -cancel 40 ns
isim force add {/lab0/c} 0 -radix bin -value 1 -radix bin -time 5 ns -repeat 10 ns -cancel 40 ns
isim force add {/lab0/b} 0 -radix bin -value 1 -radix bin -time 10 ns -repeat 20 ns -cancel 40 ns
isim force add {/lab0/a} 0 -radix bin -value 1 -radix bin -time 20 ns -repeat 40 ns -cancel 40 ns
```

- symulacje należy uruchomić poleceniem *run 60ns* w konsoli lub poprzez wybór jednego z poleceń z menu *Simulation* (*Run, Run All, Step)*



- po wykonaniu symulacji należy porównać otrzymane przebiegi czasowe z tablicą prawdy badanej funkcji

- poszczególne wymuszenia można realizować za pomocą komend widzianych/wpisywanych w oknie *Console*, istnieje możliwość stworzenia i uruchomienia własnych skryptów poprzez komendę *do*. Opis poszczególnych funkcji można uzyskać poprzez komendę *help* w konsoli.

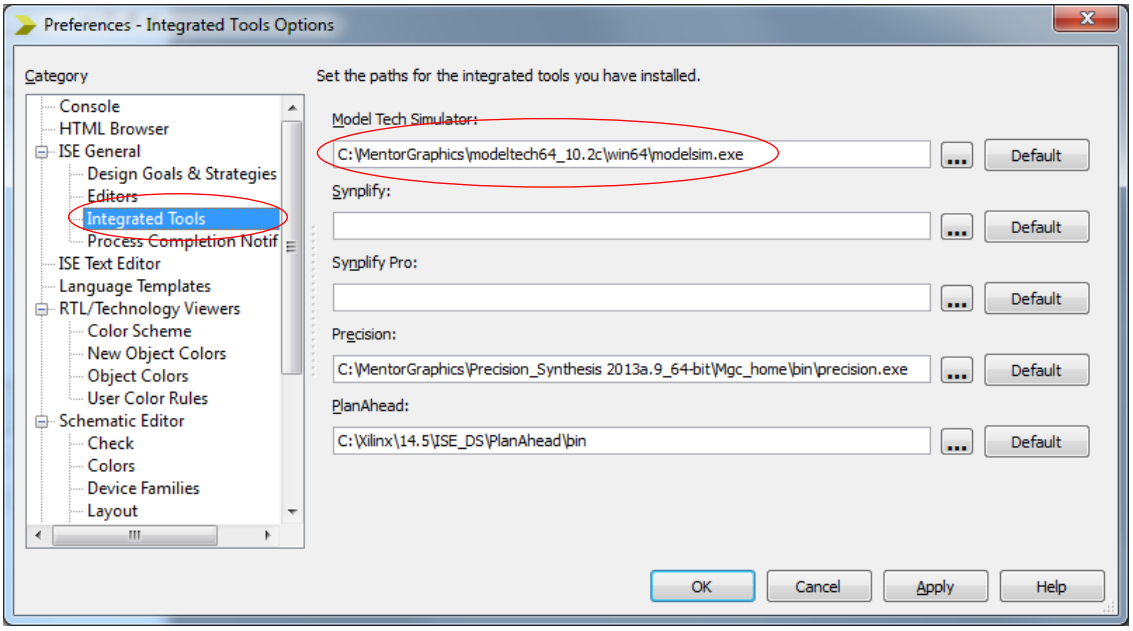
6. Symulacja behawioralna (ModelSim)

6.1. Konfiguracja symulatora

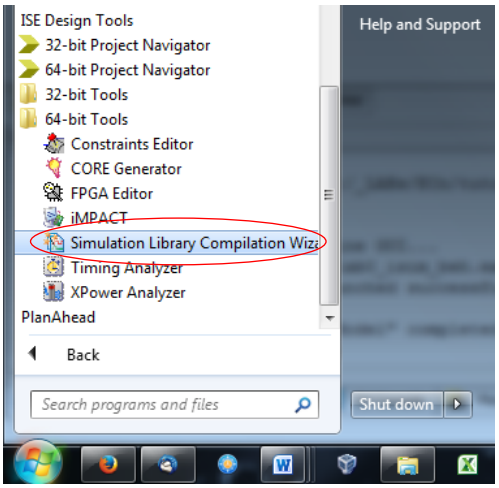
Uwaga: Poniższy fragment instrukcji, aż do zaznaczonego miejsc, dotyczy instalacji oraz przygotowania na komputerze domowym środowiska ISE WebPack wraz z symulatorem **ModelSim PE Student Edition**. Operacji tych nie należy wykonywać w laboratorium!

Program symulatora ModelSim PE Student Edition należy pobrać ze strony firmy Mentor Graphics:
http://www.mentor.com/company/higher_ed/modelsim-student-edition

- Po instalacji programu ModelSim PE Student Edition należy wskazać jego lokalizację w środowisku ISE. W tym celu z menu *Edit -> Preferences* wybieramy kategorię *ISE General -> Integrated Tools*, następnie należy podać odpowiednią ścieżkę instalacji dla symulatora *Model Tech Simulator:*.

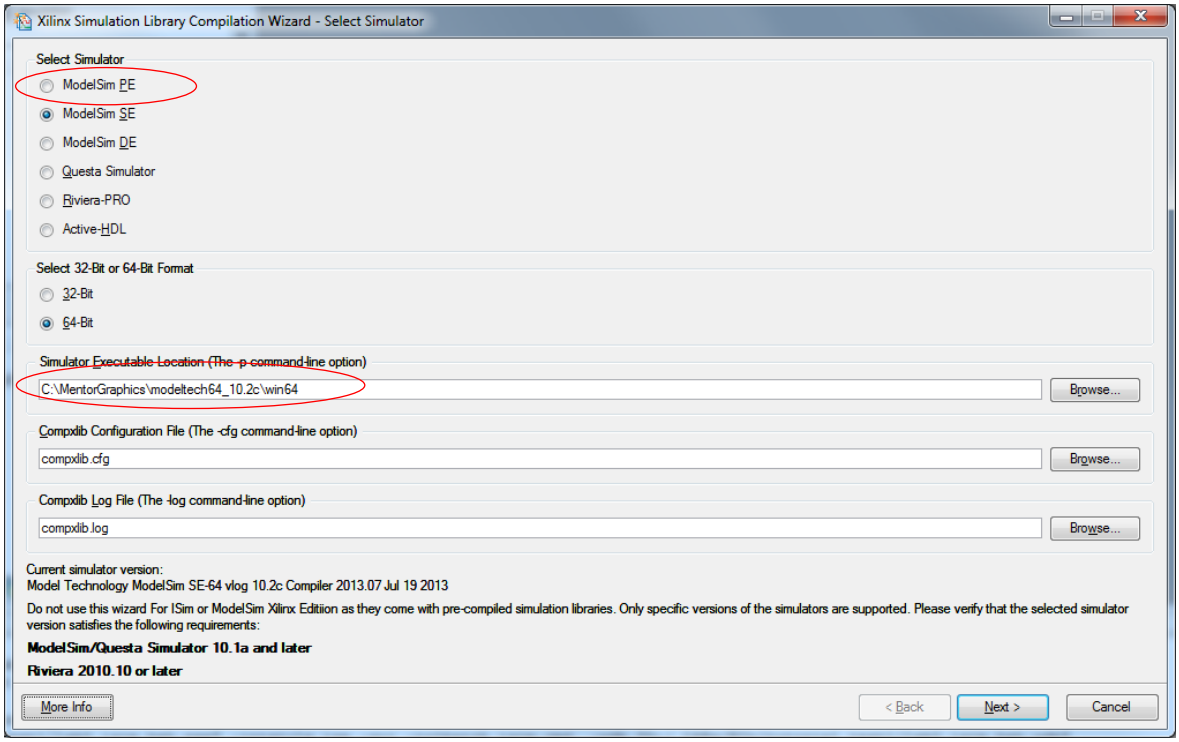


- Aby symulator poprawnie interpretował wszystkie modele producenta należy je uprzednio skompilować – **jest to operacja długotrwała**, ale wykonywana jednokrotnie podczas konfiguracji środowiska. Kompilację należy uruchomić poprzez *Simulation Library Compilation Wizard* dostępny w grupie *ISE Design Tools*.



- Po otwarciu okna programu kompilacji bibliotek (rysunek poniżej) należy kolejno wybrać odpowiedni typ stosowanego symulatora (ModelSim PE) oraz podać ścieżkę do katalogu, w którym znajduje się plik modelsim.exe (jeśli 'wizard' sam go nie odnajdzie). We wszystkich następnych oknach dialogowych należy pozostawić ustawienia domyślne i uruchomić kompilację bibliotek. Poprawnie przeprowadzona kompilacja

powinna zakończyć się oknem podsumowania, w którym mogą pojawić się ostrzeżenia (Warnings), ale nie powinno być żadnej informacji o błędach (Errors)!



- Ostatnim krokiem konfiguracji środowiska jest uzupełnienie pliku modelsim.ini informacją o położeniu skompilowanych bibliotek. W tym celu należy odszukać na dysku dwa pliki modelsim.ini:

  1. modelsim.ini znajdujący się w katalogu instalacyjnym symulatora Modelsim-PE
  2. modelsim.ini znajdujący się w katalogu instalacyjnym pakietu Xilinx ISE (podkatalog ISE_DS)

  W pliku 1 należy odszukać grupę [Library] i skopiować tam wpisy dotyczące bibliotek *unisim*, *unimacro*, *simprim*, *cpld* i *xilinxcorelib* z pliku 2. Przykładowy wygląd ścieżek poniżej:

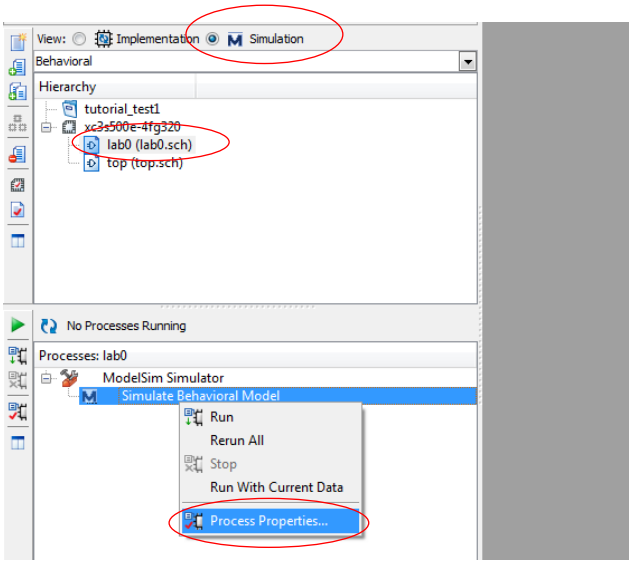
```
unisim = C:\Xilinx\14.5\ISE_DS\ISE\vhdl\mti_se\10.2c\nt64/unisim
unimacro = C:\Xilinx\14.5\ISE_DS\ISE\vhdl\mti_se\10.2c\nt64/unimacro
simprim = C:\Xilinx\14.5\ISE_DS\ISE\vhdl\mti_se\10.2c\nt64/simprim
xilinxcorelib = C:\Xilinx\14.5\ISE_DS\ISE\vhdl\mti_se\10.2c\nt64/xilinxcorelib
cpld = C:\Xilinx\14.5\ISE_DS\ISE\vhdl\mti_se\10.2c\nt64/cpld
```

Uwaga: Koniec fragmentu dotyczącego konfiguracji symulatora.

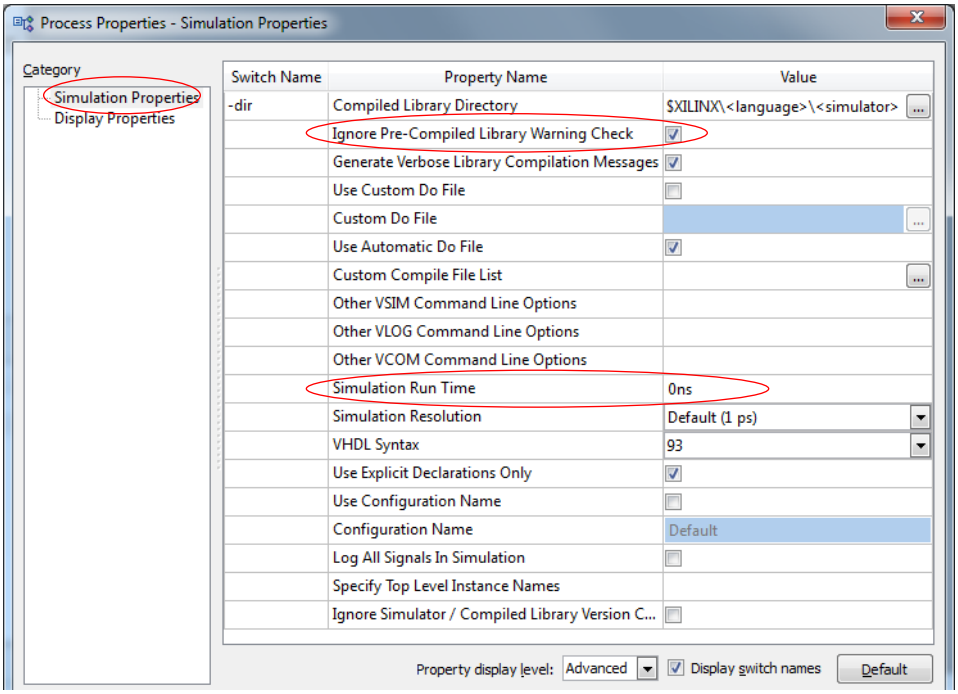
6.2. Uruchomienie symulacji

Ustawiamy parametry uruchomienia symulatora:

- w zakładce *Design* ustawiamy rodzaj widoku na *Simulation*, wybieramy plik lab0.sch i z menu w oknie procesów wybieramy opcję *Process Properties*

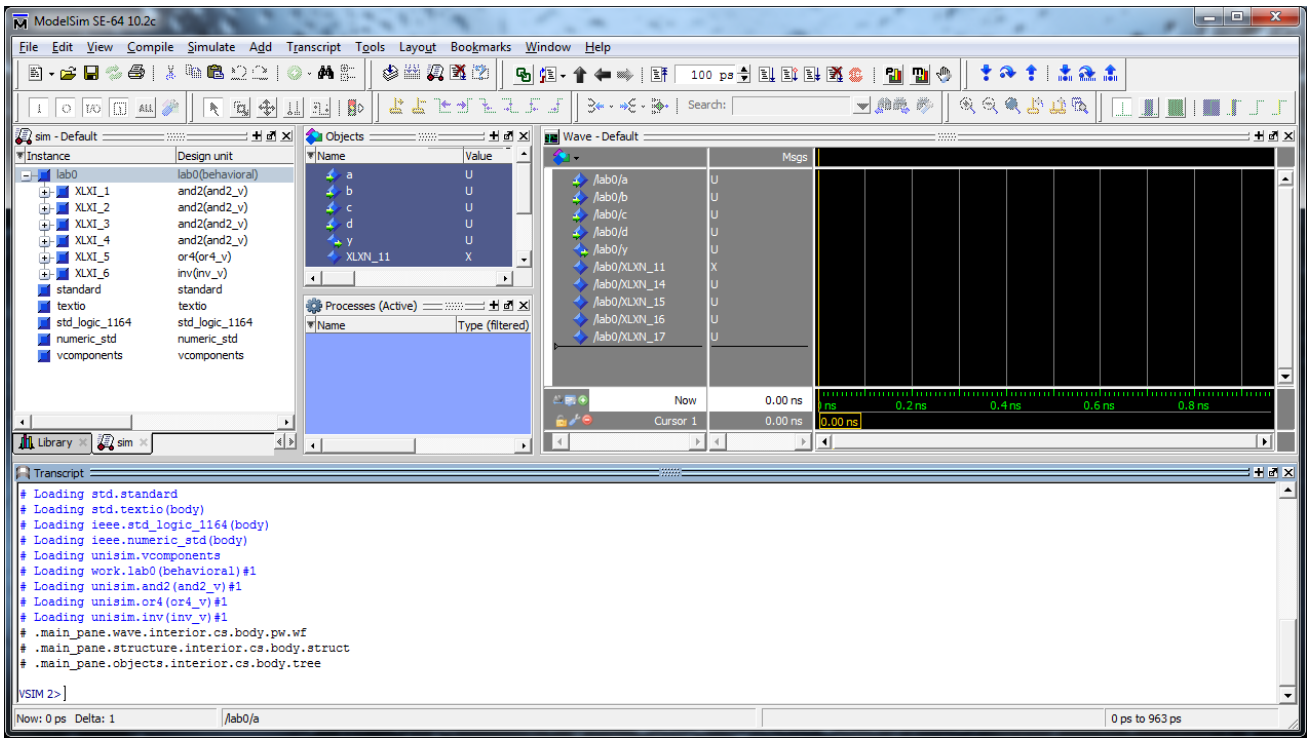


- w oknie dialogowym *Process Properties*, w zakładce *Simulation Properties* zaznaczamy opcję *Ignore Pre-Compiled Library Warning Check* oraz zmieniamy startowy czas uruchomienia symulacji na **0ns**



- w zakładce *Design* ustawiamy rodzaj widoku na *Simulation*, wybieramy plik **lab0.sch** i uruchamiamy symulator poleceniem *Simulate Behavioral Model -> Run*. Symulator jest programem innego producenta i zostanie uruchomiony w nowym oknie. Po uruchomieniu nastąpi automatyczne otwarcie projektu odpowiadającego symulowanemu plikowi lab0.sch, kompilacja niezbędnych źródeł i uruchomienie symulacji na czas zdefiniowany w opcji *Simulation Run Time*.

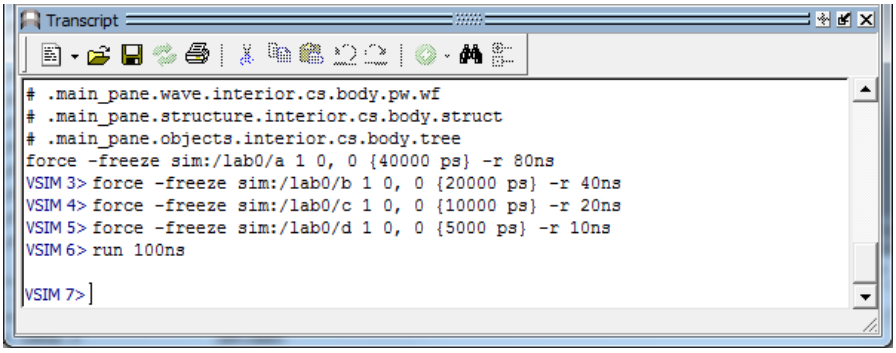
- program ModelSim zawiera następujące okna:
  - *sim* (*Workspace)*- w którym widzimy hierarchię skompilowanego projektu załadowanego do symulacji,
  - *Objects* - w którym wybieramy do wyświetlania dany sygnał układu, zaznaczając go na liście i wybierając polecenie *Add to Wave -> Selected Signals*,
  - *Wave* - zawierające przebiegi czasowe badanych sygnałów,
  - *Transcript* – (okno transkrypcji) konsola zawierająca interpreter języka Tcl.



- w oknie *Wave* istnieje możliwość realizacji wymuszeń, poprzez zaznaczenie danego sygnału i wybór jednego z poleceń z podręcznego menu:

◦ *Force*…– wymuszenie stałe o określonej długości,
◦ *NoForce*– usunięcie wymuszenia,
◦ *Clock*… – wymuszenie okresowe.

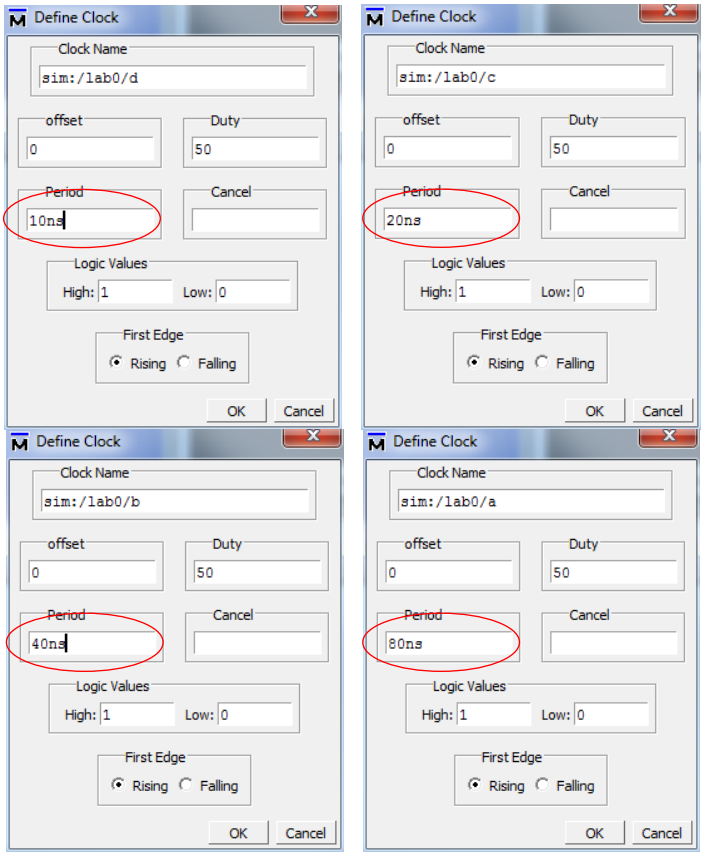
- poszczególne wymuszenia można realizować za pomocą komend widzianych/wpisywanych w oknie *Transcript*, istnieje możliwość podłączania własnych skryptów i uruchamiania ich poprzez komendę *do*. Opis poszczególnych funkcji można uzyskać poprzez komendę *help*.



Dokładne wyjaśnienie komend używanych w oknie transkrypcji można znaleźć w dokumentacji symulatora (ModelSim PE Reference Guide). Poniżej podane zostaną jedynie przykłady najczęstszego użycia.

Uwaga: we wszystkich poleceniach symulatora brak podania jednostki czasu oznacza przyjęcie domyślnej jednostki, którą jest pikosekunda *ps*.

- definiowanie wymuszeń dla symulacji lab0.sch:
  w oknie *Wave* symulatora wybieramy sygnał d i z menu kontekstowego wybieramy opcję *Clock*, następnie w oknie dialogowym ustawiamy wartość *Period* na 10ns, wartość *Cancel* można pominąć – wtedy sygnał będzie generowany w nieskończoność; w podobny sposób tworzymy wymuszenia dla pozostałych wejść za każdym razem mnożąc okres wymuszenia x2



- symulację włączamy poprzez wybór z menu *Simulate -> Run -> Run* lub wydając polecenie *run 100ns* w oknie transkrypcji

- wymuszenie sygnałów identyczne do podanego wyżej można uzyskać wykonując poniższe polecenia: (polecenie: *restart -force* powoduje restart symulatora czyli powrót do czasu symulacji równego 0ns)

```
restart -force
force d 1 0ns, 0 5ns -r 10ns
force c 1 0ns, 0 10ns -r 20ns
force b 1 0ns, 0 20ns -r 40ns
force a 1 0ns, 0 40ns -r 80ns
run 100ns
```

- w celu usprawnienia pracy i uniknięcia ponownego wprowadzania tych samych poleceń można zawrzeć je w tzw. pliku makra. Jest to plik tekstowy zawierający polecenia języka Tcl i polecenia wewnętrzne symulatora. Wszystkie polecenia z pliku makra możemy wykonać jako skrypt poprzez komendę *do* w oknie transkrypcji:

```
do <nazwa pliku makra>          np.:
do forsuj.do
```

Uwaga: Jeżeli w trakcie weryfikacji układu okaże się, że zawiera on błąd, można wrócić do środowiska ISE WebPack (edytora schematów) i poprawić błąd. W celu ponownego sprawdzenia projektu nie jest konieczne zamykanie okna symulatora. W środowisku ISE poprawiony plik (schemat) należy zapisać, a następnie wrócić do zakładki *Design*, zmienić widok na *Implementation* i z grupy *Design Utilities* wybrać opcję *View VHDL Functional Model*. Operacja ta spowoduje odświeżenie modelu wykorzystywanego przez symulator ModelSim. Następnie w programie symulatora, w oknie *Transcript*, używając klawiszy góra-dół należy odnaleźć linijkę uruchamiającą skrypt symulatora odpowiedzialny za przygotowanie środowiska symulacji. W przypadku symulacji schematu lab0.sch polecenie to wygląda jak poniżej:

```
do {lab0.fdo}
```

Należy uruchomić to polecenie. Spowoduje ono „gorący restart" symulatora, tzn. ponowne załadowanie i przygotowanie środowiska symulacji. Niestety komenda ta powoduje anulowanie wymuszeń uprzednio podanych na wejścia. Można je ponownie dodać poprzez wywołanie skryptu, w którym zachowane zostały polecenia ustawiające wymuszenia jak w powyższym przykładzie (np. `do forsuj.do`).

7. Przypisanie wyprowadzeń

Po sprawdzeniu poprawności wykonania urządzenia można przystąpić do scalania projektu z interfejsem płyty prototypowej. Najistotniejszym krokiem tego etapu prototypowania jest wybór wyprowadzeń układu scalonego, z którymi zostaną związane sygnały/porty w naszym projekcie. Wyboru tego dokonujemy na podstawie znajomości zasobów płyty i wymagań dotyczących zaprojektowanego urządzenia. Zadanie to wymaga zapoznania się z dokumentacją stosowanej płyty prototypowej.

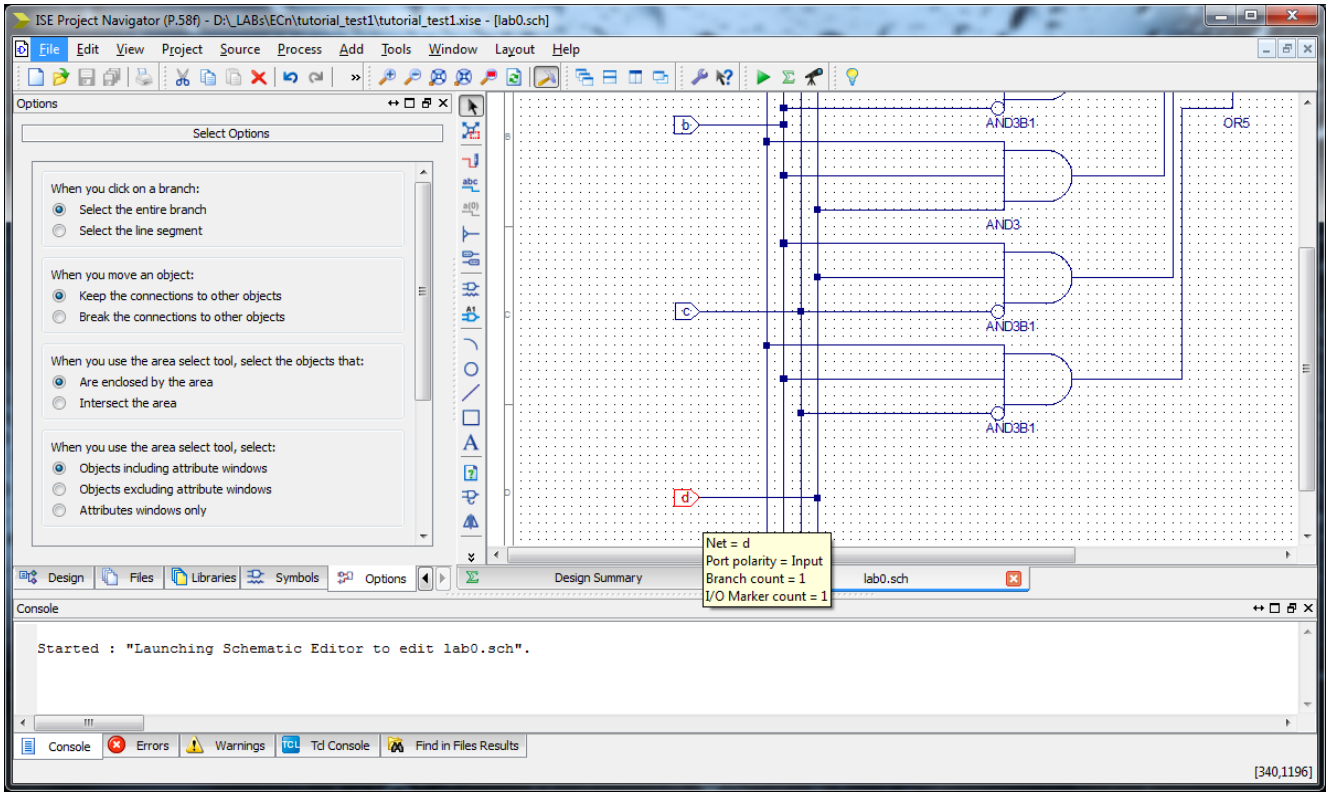
W urządzeniu kombinacyjnym opisanym schematem lab0.sch mamy 4 wejścia 1-bitowe i jedno wyjście 1-bitowe. Do prostego sterowania wejściami możemy wykorzystać przełączniki SWx lub przyciski BTNx, można też podłączyć źródła wymuszeń, takiej jak generatory przebiegów prostokątnych, poprzez złącza JA-JD. Do kontroli odpowiedzi układu można wykorzystać diody LDx, wyświetlacz multipleksowany DISP1 lub analizator stanów logicznych podłączony do złącz JA-JD. Wyprowadzenia w złączach JA-JD mogą zostać użyte dowolnie: jako wejścia, wyjścia lub porty dwukierunkowe. Zależy to od kierunkowości portu na schemacie.

Dla szybkiej kontroli działania układu wybierzemy podłączenie przełączników **SW3,SW2,SW1,SW0 do wejść a,b,c,d** oraz diody **LD7 do wyjścia y**.

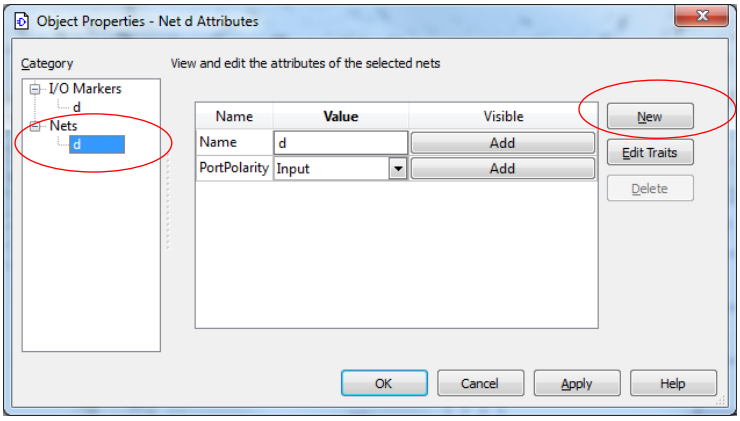
Przypisanie wyprowadzeń można wykonać trzema drogami:
- bezpośrednio na schemacie,
- w pliku konfiguracyjnym UCF,
- w kodzie modelu VHDL.

W poniższej instrukcji zostanie przedstawiony pierwszy z wymienionych sposobów.

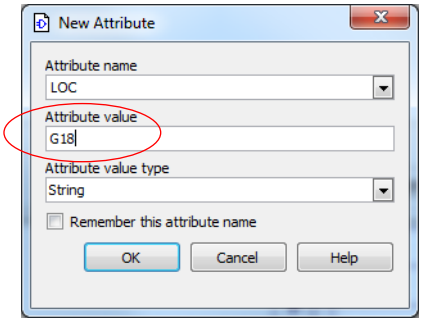
- otwórz schemat lab0.sch
- zaznacz marker portu **d** i z menu kontekstowego wybierz polecenie *Object Properties*



- w oknie dialogowym wybierz polecenie *New* aby dodać nowy atrybut dla sieci **d**



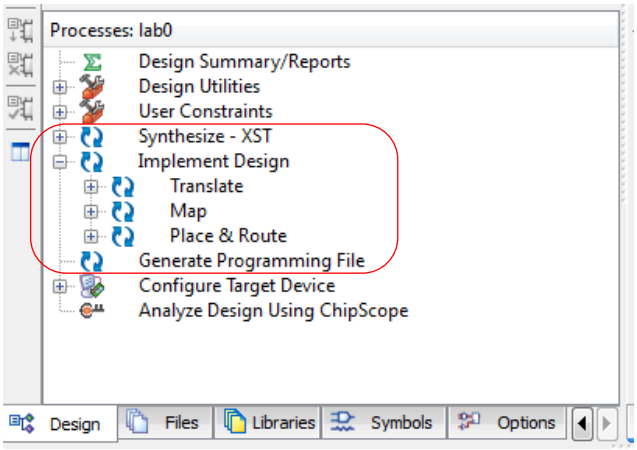
- wprowadź nazwę atrybutu **LOC** oraz numer wyprowadzenia wg dokumentacji lub opisów na płycie Nexys2 (dla przełącznika SW0 będzie to G18)



- powtórz powyższe czynności dla wejść c,b,a (SW1=H18, SW2=K18, SW3=K17) i wyjścia y (LD7=R4).
- zapisz zmiany wykonane w schemacie.

8. Implementacja

Implementacja projektu składa się z wielu etapów, które nie będą szczegółowo omawiane w tej instrukcji. Najogólniej mówiąc, proces ten polega na zamianie plików wejściowych w postaci graficznej lub kodu języka opisu sprzętu na netlisty zawierające elementy logiczne (bramki, przerzutniki, małe pamięci itp.). W kolejnym kroku następuje ich umieszczenie na powierzchni układu scalonego FPGA i automatyczne trasowanie połączeń. Ostatnim etapem jest przygotowanie pliku konfiguracyjnego dla konkretnego modelu układu scalonego. Cykl ten znajduje odzwierciedlenie w nazwach poszczególnych procesów projektowych w środowisku ISE: synteza, translacja, mapowanie, trasowanie, przygotowanie pliku do programowania (patrz rysunek obok).

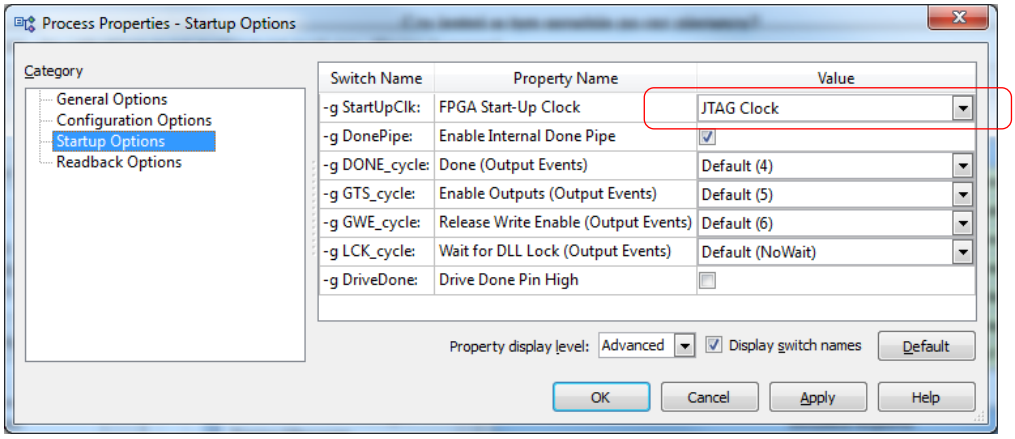


Wymienione wyżej procesy dostępne są tylko w widoku *Implementation* dla źródła projektowego oznaczonego jako *Top Module* (charakterystyczna ikona przy pliku w oknie hierarchii projektu).

8.1. Generacja pliku konfiguracyjnego

Jeśli w wykonanym schemacie głównym nie ma błędów można przystąpić do implementacji projektu. Z uwagi na prostotę wykonywanych projektów wszystkie opcje syntezy i implementacji w systemie ISE można pozostawić w ustawieniach domyślnych.

- przejdź do zakładki *Design*, przełącz widok na *Implementation* i zaznacz źródło lab0.sch (powinno być wyróżnione ikoną , jeśli tak nie jest wybierz polecenie z menu *Source -> Set as Top Module*)
- zaznacz proces *Generate Programming File* w oknie procesów i z menu kontekstowego otwórz okno ustawień poleceniem *Process Properties...*
- w oknie dialogowym wybierz kategorię *Startup Options* i ustaw *FPGA Start-Up Clock* na *JTAG Clock*, zatwierdź zmiany i zamknij okno

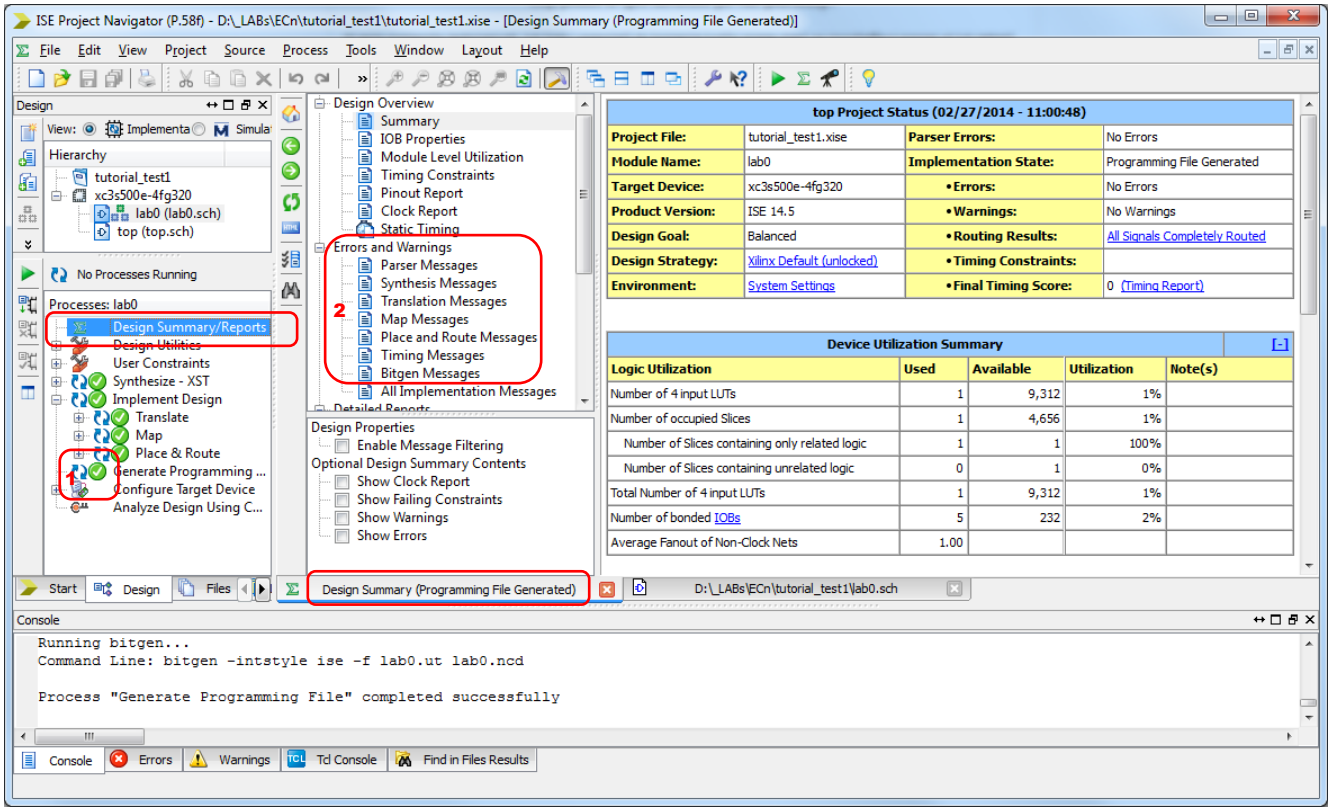


- zaznacz proces *Generate Programming File* w oknie procesów i wygeneruj plik bitowy projektu poleceniem *Run* z menu kontekstowego.

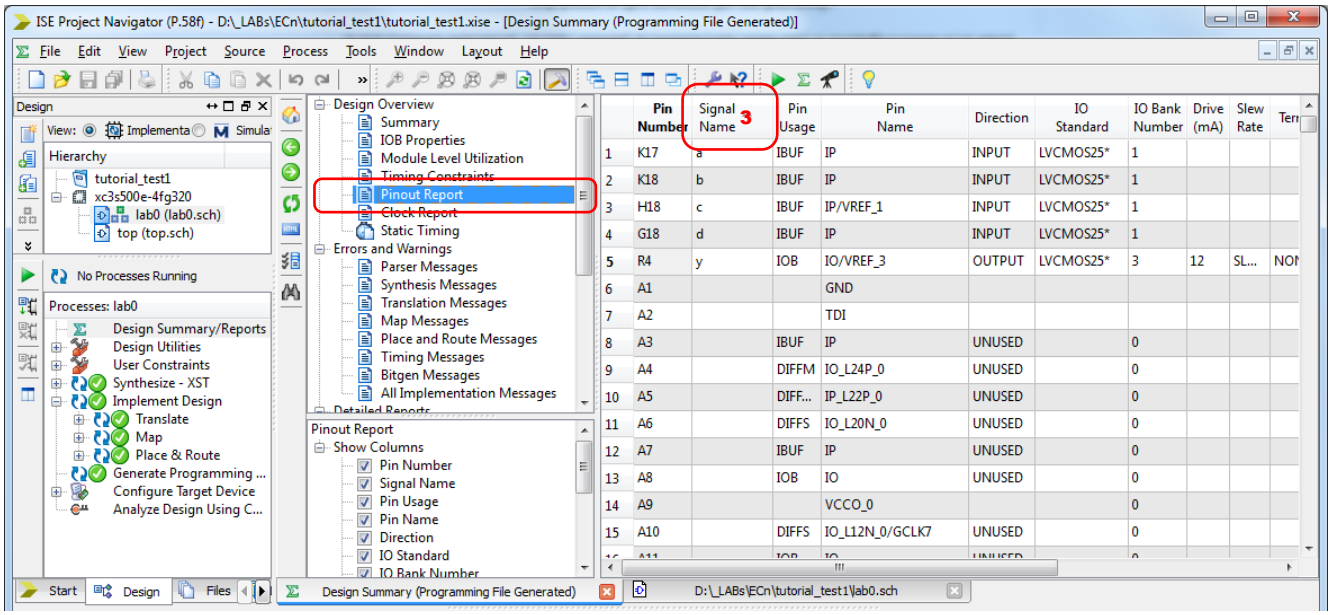
Uwaga: po pomyślnym wykonaniu tej operacji proces tworzenia urządzenia można uznać za zakończony; wszystkie brakujące kroki cyklu projektowania zostaną wykonane automatycznie. Po zakończeniu implementacji zostanie utworzony plik bitowy do konfigurowania matrycy FPGA – nazwa tego pliku jest zawsze taka sama jak pliku oznaczonego *Top Module* z rozszerzeniem .bit (w naszym przypadku **lab0.bit**).

8.2. Weryfikacja poprawności

Podczas cyklu projektowania generowany jest zbiór raportów dotyczących kolejnych etapów wykonania projektu. Są one istotne na etapie wykrywania i usuwania błędów w wykonywanym urządzeniu. Dostęp do raportów ułatwia zakładka podsumowania projektu (*Design Summary*) otwierana poleceniem *Design Summary/Reports* z okna procesów.



Zielone ikony przy kolejnych procesach (1) oznaczają bezproblemowe zakończenie tych operacji (bez ostrzeżeń i błędów). Informacje o problemach występujących w kolejnych krokach wykonania projektu można przeglądać w postaci tekstowej poprzez listę *Errors and Wrnings* (2). Błędne przypisanie wyprowadzeń nie jest, przez środowisko ISE, traktowane jako błąd projektowy. Dlatego weryfikacji poprawnego przypisania należy dokonać samodzielnie w raporcie *Pinout Report*.



Uwaga: sortując zawartość raportu względem zdefiniowanych przez użytkownika nazw portów (3) można skontrolować wszystkie wykonane w danej implementacji przypisania wyprowadzeń. Zależność między kolumnami *Pin Number -> Signal Name* powinna, jako pierwsza, zostać skonfrontowana z dokumentacją płyty prototypowej w przypadku pojawienia się problemów z działaniem urządzenia.

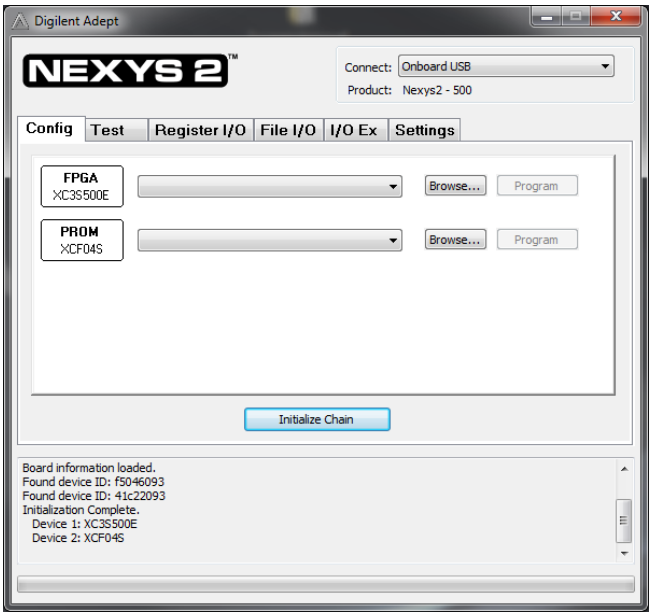
9. Konfiguracja

Konfigurację (załadowanie pliku .bit do układu FPGA) można przeprowadzić przy pomocy dwóch różnych programów. W zależności od posiadanej wersji oprogramowania ISE można wykorzystać iMPACT (jest częścią pakietu ISE) lub program Adept (narzędzie firmy Digilent – producenta płyty prototypowej).
Uwaga: wybór oprogramowania skonsultuj z prowadzącym zajęcia.

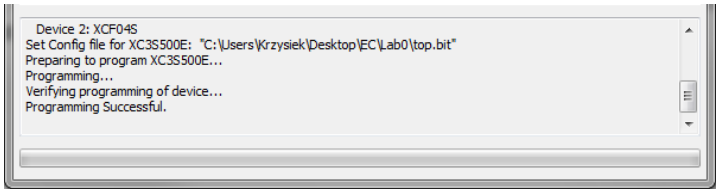
9.1. Digilent Adept

Posiadając plik .bit do programowania matrycy można przystąpić do uruchomienia urządzenia. W tym celu należy podłączyć kabel USB do płyty Nexys2 (stanowi on również zasilanie płyty). Po przygotowaniu stanowiska pomiarowego można przystąpić do operacji konfigurowania układu FPGA.

- uruchom program Adept



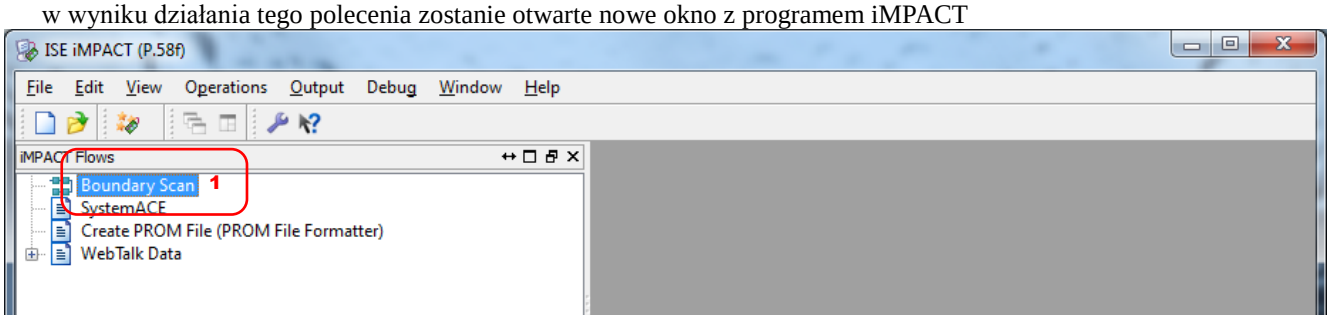
- wybierz z menu *Connect: Onboard USB*, powinien zostać wykryty *Product: Nexys2 – 500*
- przy poprawnej identyfikacji (wykryciu układów znajdujących się na płycie prototypowej przez oprogramowanie) powinniśmy zobaczyć dwa układy: FPGA i PROM (rysunek powyżej).
- za pomogą przycisku *Browse* przy układzie FPGA, wybierz wygenerowany plik lab0.bit (jest w katalogu głównym wykonywanego projektu) i wgraj go do układu FPGA XC3S500E poprzez przyciśnięcie przycisku *Program*
- poprawne wgranie pliku do układu zostanie podsumowane poniższym komunikatem.



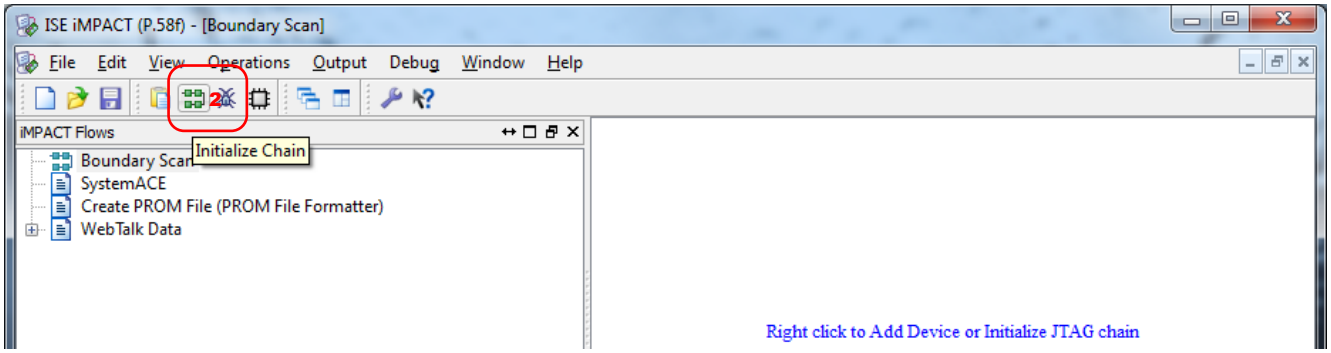
9.2. iMPACT

Posiadając plik .bit do programowania matrycy można przystąpić do uruchomienia urządzenia. W tym celu należy podłączyć kabel USB do płyty Nexys2 (stanowi on również zasilanie płyty). Po przygotowaniu stanowiska pomiarowego można przystąpić do operacji konfigurowania układu FPGA.

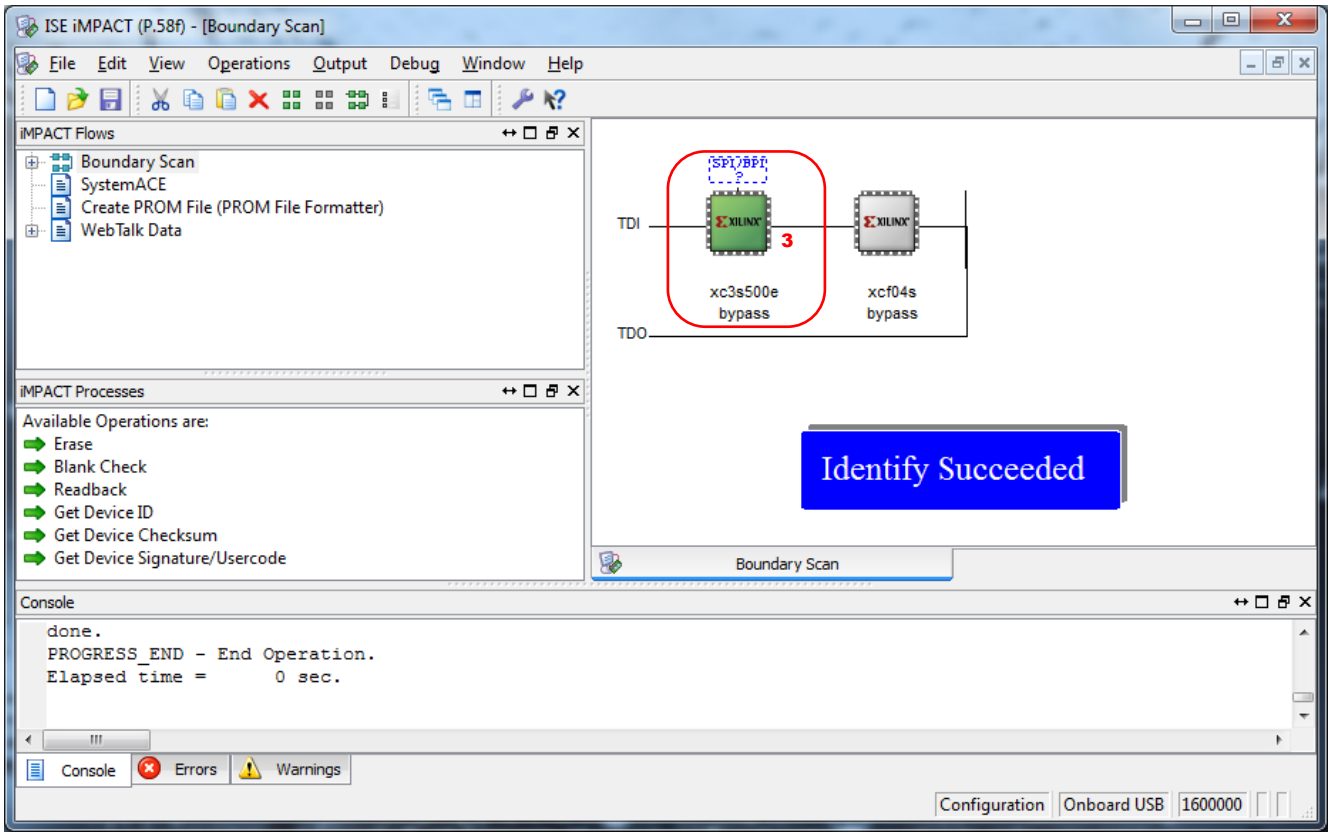
- rozwiń grupę *Configure Target Device* w oknie procesów i uruchom polecenie *Manage Configuration Projec (iMPACT)* w wyniku działania tego polecenia zostanie otwarte nowe okno z programem iMPACT



- wybierz tryb programowania *Boundary Scan* (1)



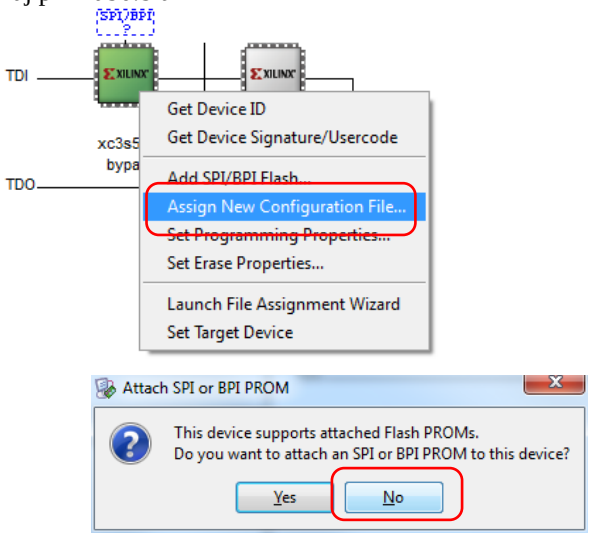
- dokonaj inicjalizacji łańcucha JTAG - *Initialize Chain* (2)



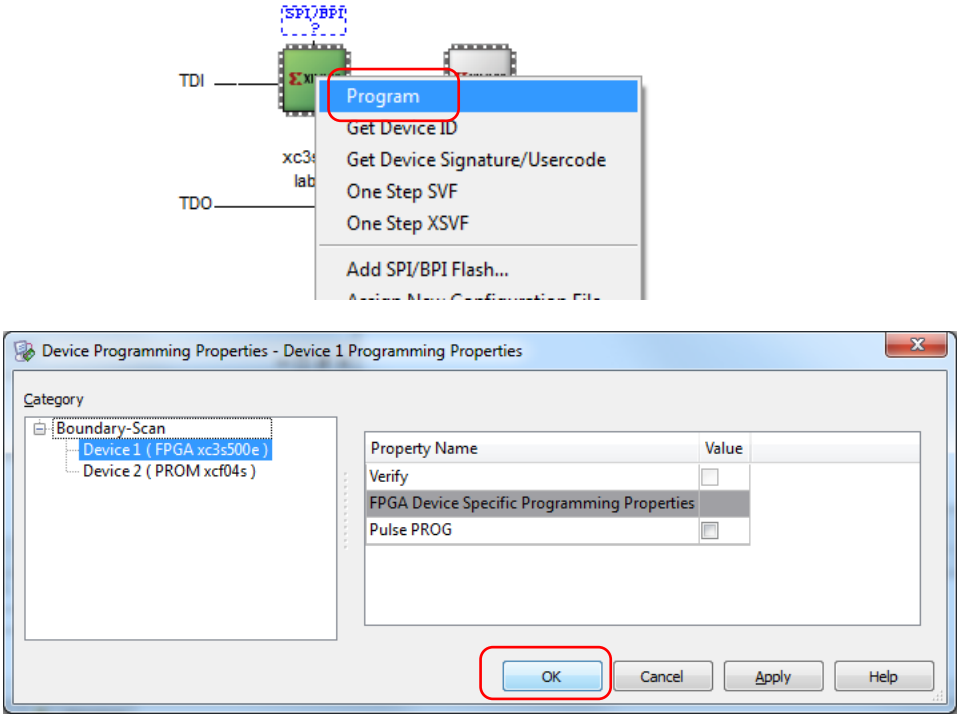
- poprawnym efektem tych działań na płycie Nexys2 jest wykrycie dwóch układów scalonych, z których pierwszy jest matrycą FPGA (3)

Uwaga: w laboratorium zawsze konfigurowany będzie tylko ten jeden układ.

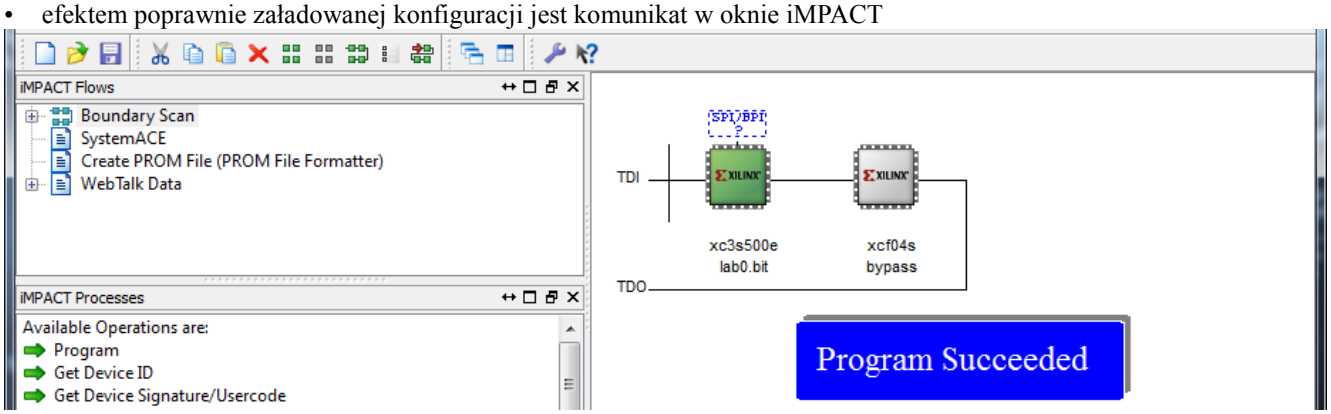
- w celu przypisania pliku konfiguracji, z menu kontekstowego układu FPGA (3) wybierz polecenie *Assign New Configuration File* i odszukaj plik lab0.bit



- z menu kontekstowego układu FPGA (3) wybierz polecenie *Program*



- efektem poprawnie załadowanej konfiguracji jest komunikat w oknie iMPACT



10. Testowanie urządzenia

Testowanie układu lab0 na płycie Nexys2 sprowadza się do ustawienia wszystkich kombinacji wymuszeń z tablicy prawdy przy pomocy kluczy SW3-SW0 (pamiętając o tym, że SW0 jest najmłodszym bitem) i kontrola świecenia diody LD7 (zapalona = ‘1’ logiczne, zgaszona=’0’ logiczne).

Taki sposób weryfikacji jest czasochłonny (szczególnie dla funkcji wielu zmiennych) i trudny do udokumentowania. Dużo wygodniejszym podejściem jest zastosowanie generatora wymuszeń po stronie wejściowej i analizatora stanów logicznych po stronie odpowiedzi. Podejścia takie wiążą się z modyfikacją projektu i będą analizowane w następnych punktach instrukcji.

10.1. Podłączenie analizatora

W celu podłączenia analizatora stanów logicznych należy sygnały, które zamierzamy badać, wyprowadzić na porty JA-JD. W naszym przypadku (tylko 5 sygnałów 1-bitowych) wystarczy użycie złącza JA.

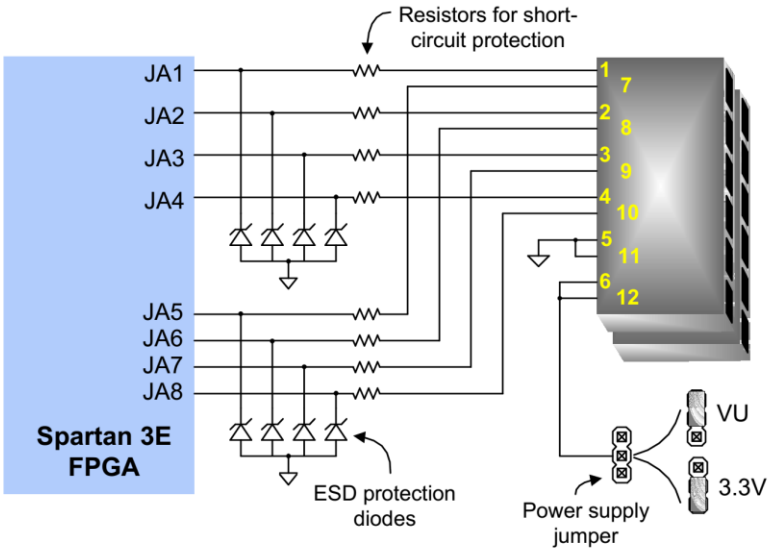


Figure 23: Nexys2 Pmod connector circuits

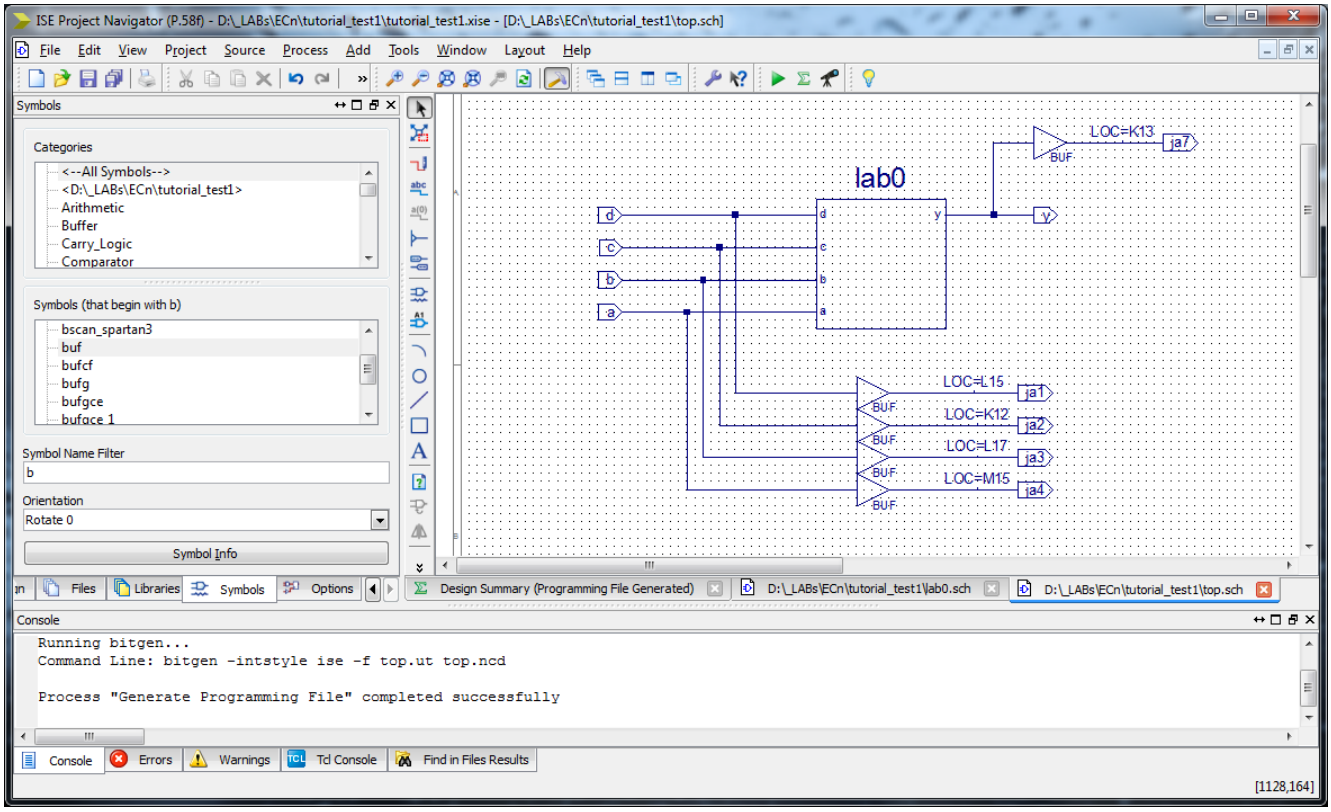
Table 3: Nexys2 Pmod Connector Pin Assignments							
Pmod JA		**Pmod JB**		**Pmod JC**		**Pmod JD**	
JA1: L15	JA7: K13	JB1: M13	JB7: P17	JC1: G15	JC7: H15	JD1: J13	JD7: K14[1]
JA2: K12	JA8: L16	JB2: R18	JB8: R16	JC2: J16	JC8: F14	JD2: M18	JD8: K15[2]
JA3: L17	JA9: M14	JB3: R15	JB9: T18	JC3: G13	JC9: G16	JD3: N18	JD9: J15[3]
JA4: M15	JA10: M16	JB4: T17	JB10: U18	JC4: H16	JC10: J12	JD4:; P18	JD10: J14[4]

(fragment dokumentacji płyty Nexys2)

Uwaga: w celu rozbudowy struktury projektu najlepiej posługiwać się podejściem modułowym. Nie należy zmieniać/rozbudowywać raz wykonanych i przetestowanych bloków (schematów). Sugerowane jest raczej ponowne ich wykorzystanie jako bloków w rozbudowanej hierarchii projektu. W złożonych projektach przypisanie wyprowadzeń powinno następować na schemacie poziomu głównego lub w pliku UCF. Atrybut LOC może być przypisany tylko do portu lub sygnału, który nie jest oddzielony od portu zewnętrznego żadnymi elementami logicznymi typu bramki czy przerzutniki.

Sposób modyfikacji struktury projektu do podłączenia analizatora:

- dodaj nowe źródło projektowe typu schemat o dowolnej nazwie (np. top.sch)
- umieść w tak utworzonym schemacie symbol lab0 utworzony w punkcie 4.2 instrukcji
- dodaj dodatkowe wyprowadzenia sygnałów mierzonych na złącze JA (numeracja wg Table3: Nexys2 Pmod Connector) (należy wyprowadzić wszystkie wejścia i wyjścia modułu lab0)



Na schemacie użyto elementów BUF do oddzielenia sieci wejściowej o nazwie d od sieci wyjściowej o nazwie ja1 itd. Jest to spowodowane ograniczeniami edytora schematów. BUF nie zmienia wartości logicznej sygnału (jest tożsamością) i jest usuwany z netlisty na etapie mapowania. Dzięki temu na poziomie schematu można przypisać różne wyprowadzenia dla tej samej sieci logicznej.
Na schemacie top.sch wymieniono tylko numery wyprowadzeń dla nowopowstałych portów. Porty a,b,c,d,y powinny zostać przypisane do numerów wyprowadzeń jak w punkcie 7 instrukcji.

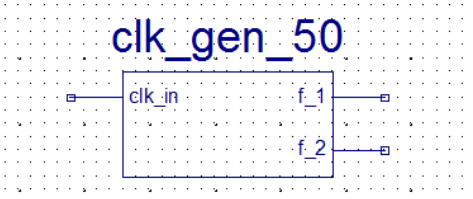
Uwaga: Umieszczenie markera portów (*I/O Marker*) na schemacie oznaczonym jako *Top Module* oznacza, że dany sygnał będzie pobierany/wystawiany na zewnątrz układu scalonego.

- po zapisaniu zmian w schemacie top.sch oznacz go jako moduł implementowany (*Set as Top Module* z menu kontekstowego w oknie *Hierarchy/Implementation*)
- wykonaj nowy plik do programowania (Generate Programming File)
- zweryfikuj poprawność przypisani w raporcie Pinout Report
- podłącz 5 kanałów analizatora stanów do portu JA (wg dokumentacji płyty prototypowej)
- podłącz zasilanie płyty Nexys2
- załaduj top.bin na platformę Nexys2 (iMPACT lub Adept)
- korzystając z analizatora w trybie 'biegnącej' podstawy czasu obserwuj przebiegi czasowe dla całej tablicy prawdy F0/4

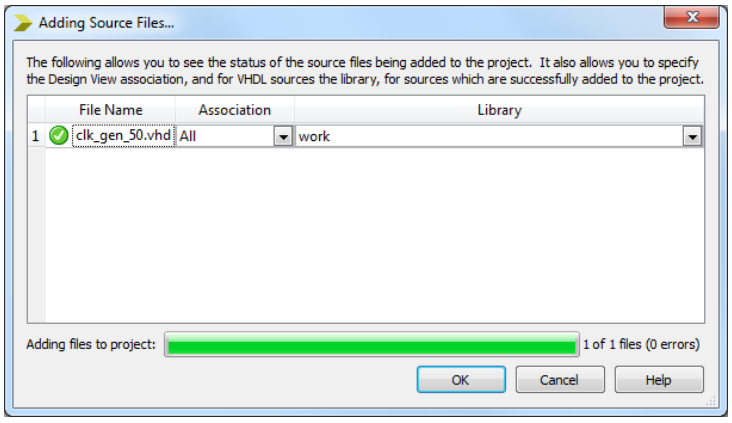
11. Wykorzystanie bloków IP

Na platformie Nexys2 wbudowany jest generator sygnału zegarowego o częstotliwości 50MHz (pin B8). Do celów testowania prototypów wykorzystywane są często dużo niższe częstotliwości pracy, w zakresie pojedynczych Hz i kHz. W tym punkcie instrukcji pokazany zostanie sposób użycia dzielnika częstotliwości opisanego w języku VHDL. Użytkownik nie musi znać składni języka – wystarczająca jest znajomość podstaw elektroniki cyfrowej.

Moduł dzielnika częstotliwości clk_gen_50 dzieli wejściowy sygnał clk_in (B8) o częstotliwości 50MHz na częstotliwość f_1 oraz f_2. Domyślne wartości tych częstotliwości to odpowiednio ~1,5Hz i ~200kHz. Mogą być także modyfikowane w kodzie VHDL.



- pobierz z serwera plik **clk_gen_50.vhd** i zapisz go w katalogu głównym projektu
- z menu *Project* wybierz polecenie *Add Source* i dodaj clk_gen_50.vhd do projektu w asocjacji *All* (oznacza to, że moduł może być używany zarówno do symulacji jak i implementacji)

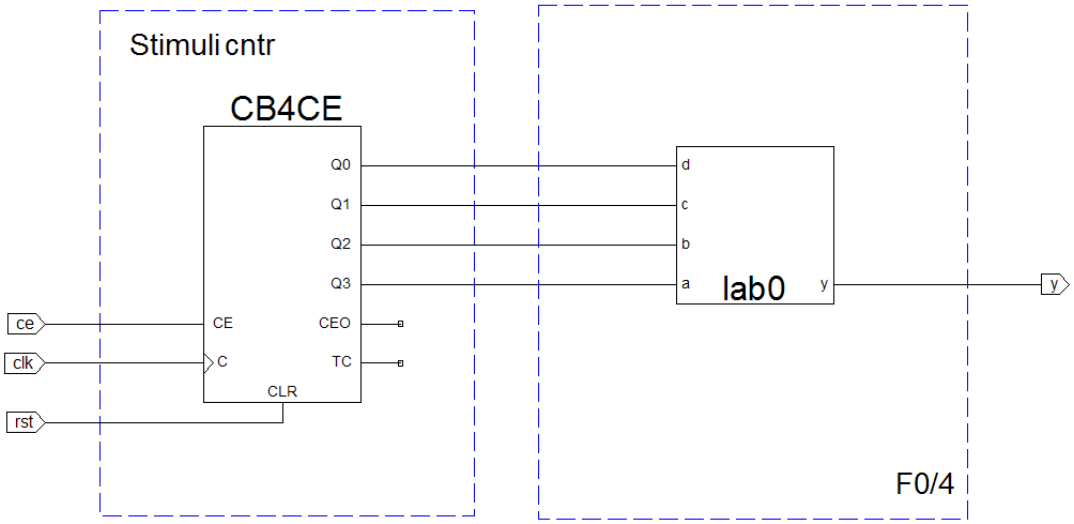


- utwórz symbol dla dodanego źródła projektowego jak w punkcie 4.2 (nowy symbol powinien pojawić się w zakładce *Symbols* w kategorii lokalnej projektu)

- wykonaj symulację behawioralną dodanego elementu **clk_gen_50**, zakładając wymuszenie o okresie 20ns i czas symulacji 2sec; pomierz okresy sygnałów na wyjściach f_1 i f_2 korzystając z dwóch kursorów w oknie *Wave* symulatora.

12. Zadania

A) 1. Korzystając z gotowego modułu licznika binarnego CB4CE (dostępny w zakładce *Symbols*) oraz wolniej zmiennego wyjścia elementu **clk_gen_50** zbuduj generator sygnałów wymuszających dla badania funkcji 4-zmiennych.
2. Wykonaj nowy schemat **top_gen_LED.sch** pozwalający na obserwację **wejść i wyjść lab0** na diodach LDx z automatyczną generacją wymuszeń.
3. Dokonaj symulacji behawioralnej i implementacji urządzenia.

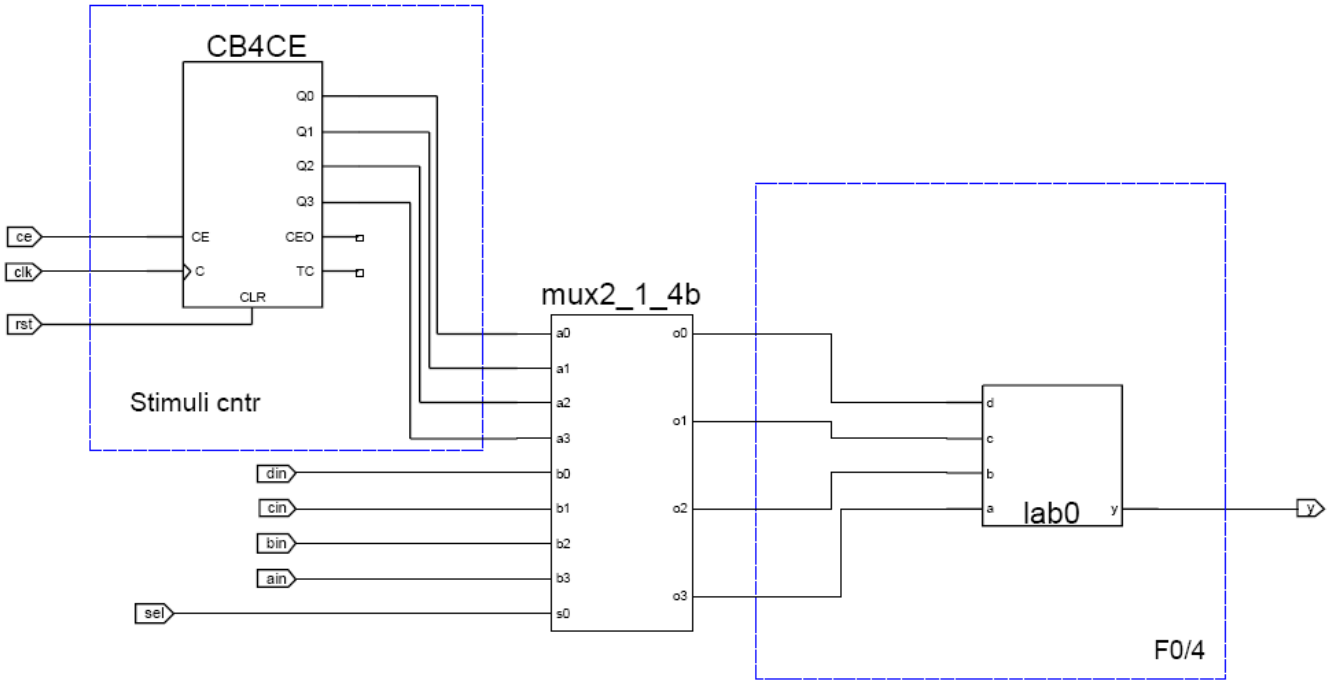


(przykład podłączenia układu generacji)

B) 1. Korzystając z wyników zadania A oraz elementów M2_1 wprowadź modyfikację realizowanego urządzenia pozwalającą na alternatywne podawanie wymuszeń na wejścia a,b,c,d (możliwość podawania wymuszeń z kluczy SWx lub z wyjścia licznika CB4CE).

2. Wykonaj nowy schemat **top_gen_LA.sch** pozwalający na obserwację **wejść i wyjść lab0** przy pomocy analizatora stanów logicznych przy automatycznej generacji wymuszeń.

3. Wykonaj implementację urządzenia i przedstaw wyniki na ekranie analizatora.



(przykład podłączenia układu z zadania B1,
element mux2_1_4b to multiplekser 4-bitowy 2x1 zbudowany z elementów M2_1 - poniżej)