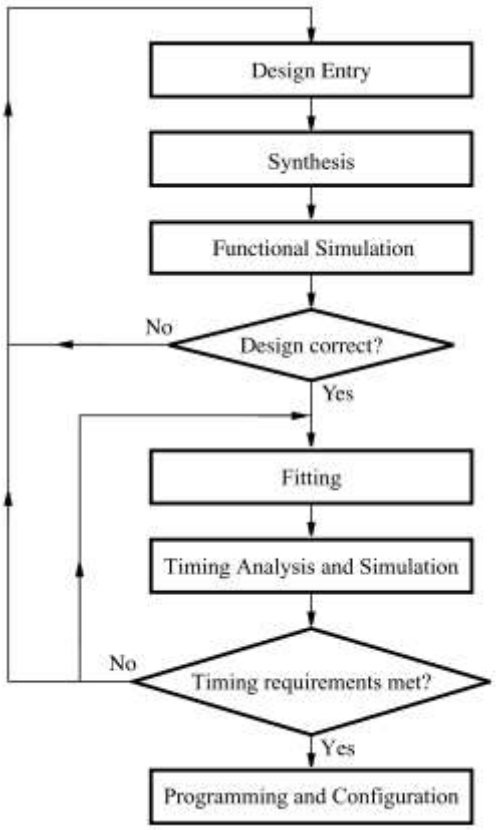
Instrukcja wykonania symulacji Verilog/VHDL oraz programowania układów MAX 10 w środowisku Quartus Prime Lite Edition (v17.0).

Wstęp

Niniejszy opis wykonania projektu układu cyfrowego dotyczy użycia języków HDL (Verilog i VHDL), oprogramowania Quartus Prime oraz układów FPGA firmy Intel z rodziny MAX 10 typu 10M08DAF256C8GES. Układ ten jest umieszczony na płytce uruchomieniowej MAXimator produkcji firmy KAMAMI będących na wyposażeniu laboratorium. W trakcie laboratorium wykorzystywane będą następujące zasoby:

- płytka uruchomieniowa MAXimator wraz z umieszczoną na niej płytką rozszerzeniową MAXimator Expander oraz programatorem JTAG zgodnym ze USB Blaster,
- oprogramowanie CAD (ang. Computer Aided Design) Quartus Prime Lite Edition firmy Intel,
- kable micro - USB (do programatora i zasilania płytki uruchomieniowej),
- zasilacz 5V DC.

Dokumentacje do ww. zasobów dostępne są w postaci linków WWW na stronie przedmiotu; http://www.ue.eti.pg.gda.pl/~bpa/jmis/jmis.html.



Rys. 1. Typowy przebieg projektowania układu cyfrowego z wykorzystaniem narzędzi CAD [1].

Oprogramowanie CAD umożliwia łatwe zaprojektowanie układu cyfrowego i zaprogramowanie funkcjonalności w cyfrowym układzie programowalnym typu CPLD (ang. Complex Programmable Logic Device) lub FPGA (ang. Field Programmable Gate Array).

Typowy przebieg projektowania cyfrowego układu programowalnego przedstawiony jest na rys. 1 [1].

Uruchomienie komputera i aplikacji CAD

W sali laboratorium EA308 w czasie włączania komputera należy wybrać system Linux. Następnie należy zalogować się jako użytkownik ***s_jmis***. Hasło do tego konta poda prowadzący zajęcia. Po zalogowaniu należy uruchomić terminal (poprzez kliknięcie prawym klawiszem myszki na pulpicie a następnie wybranie menu ***Open in Terminal***). Następnie przy użyciu komend wydawanych w terminalu należy przygotować środowisko do pracy. Wykonujemy kolejno poniższe polecenia, potwierdzając każde wpisaną komendę klawiszem ***Enter***:

1) wejście do katalogu domowego z dowolnego innego (katalog domowy dla użytkownika ***s_jmis*** jest umieszczony tu: ***/home2/s_jmis/***):

```
cd
```

2) utworzenie katalogu roboczego, w którym będą zapisywane projekty (tą czynność wykonujemy jednokrotnie, po pierwszym zalogowaniu do komputera):

```
mkdir designs
```

3) wejście do katalogu roboczego:

```
cd designs
```

4) skonfigurowanie środowiska Quartus Prime v 17.0 (to wykonujemy przed uruchomieniem oprogramowania po każdym otworzeniu terminala w którym chcemy uruchomić dane oprogramowanie):

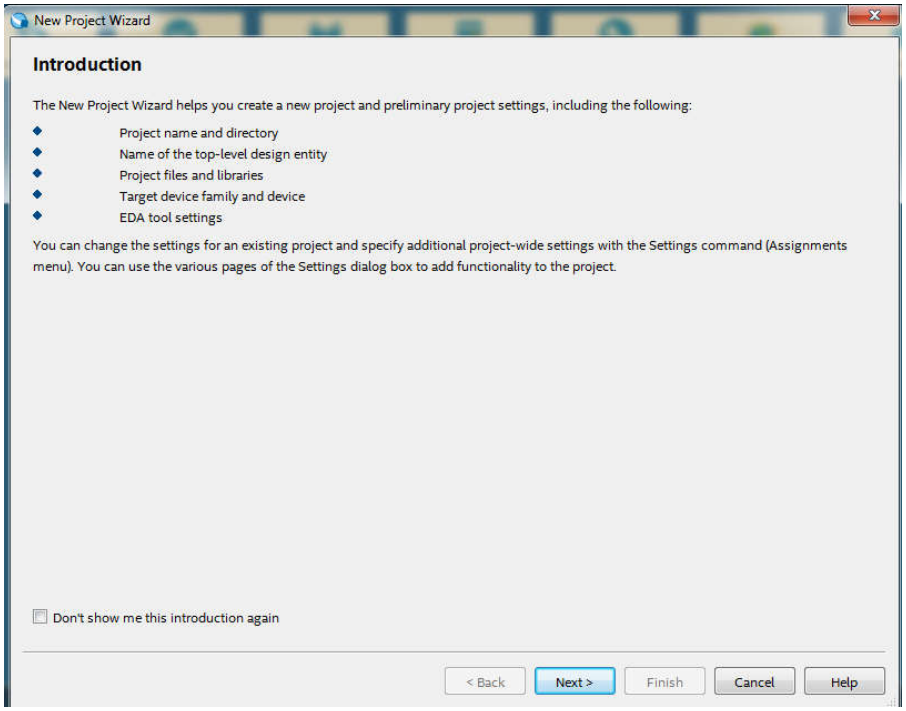
```
source /intelFPGA_lite/quartus17.csh
```

5) uruchomienie oprogramowania Quartus Prime v 17.0:

```
quartus &
```

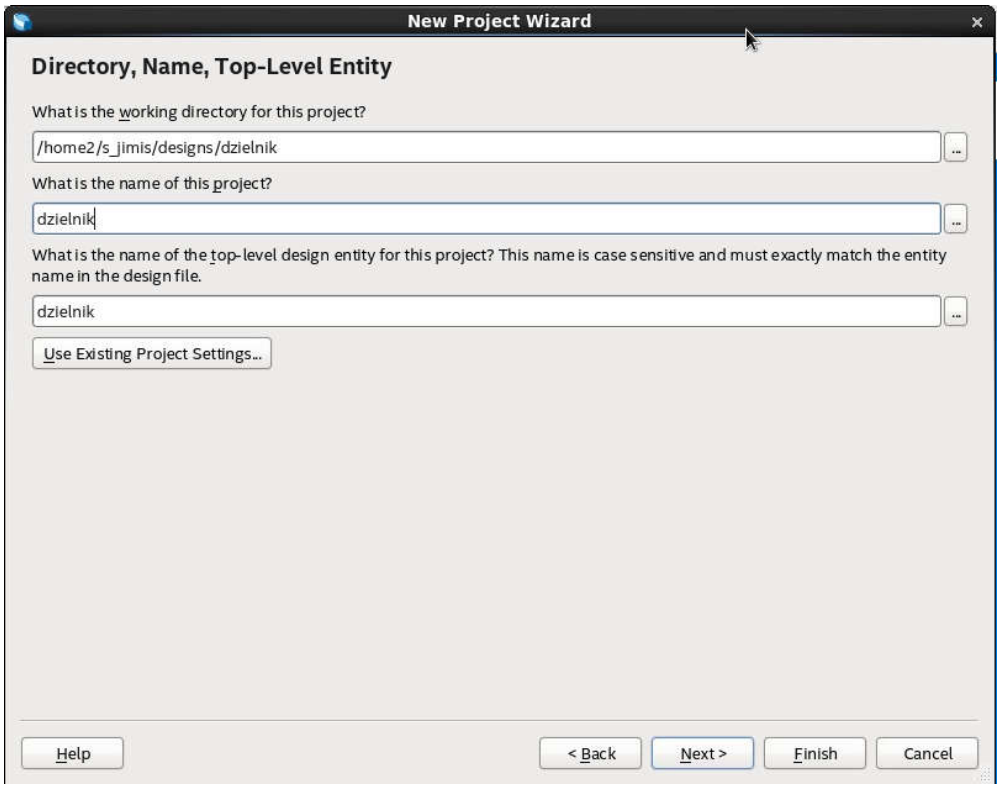
Utworzenie nowego projektu

Projekt w oprogramowaniu ***Quartus Prime*** jest traktowany jako zestaw plików umieszczonych we wspólnym katalogu, tak więc aby utworzyć nowy projekt należy stworzyć pusty katalog do którego będą zapisywane pliki projektowe. Wybieramy menu ***File → New→ New Quartus Prime Project***, powinien otworzyć się kreator nowego projektu jak na rys. 2. poniżej.



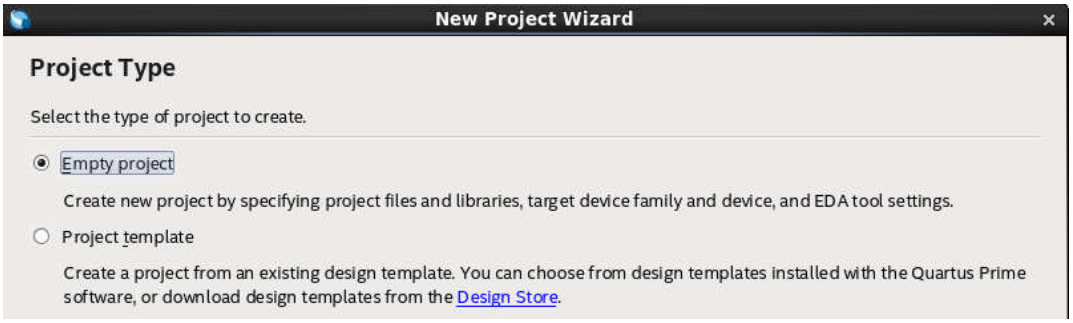
Rys. 2. Kreator nowego projektu.

Następnie wybieramy nazwę katalogu w którym projekt ma być zapisany, nazwę projektu i nazwę bloku o najwyższej hierarchii (top-level). Przykład nadania nazwy ***dzielnik*** zarówno do katalogu projektu, nazwy projektu jak i modułu o najwyższej hierarchii przedstawiono na rys. 3 poniżej.



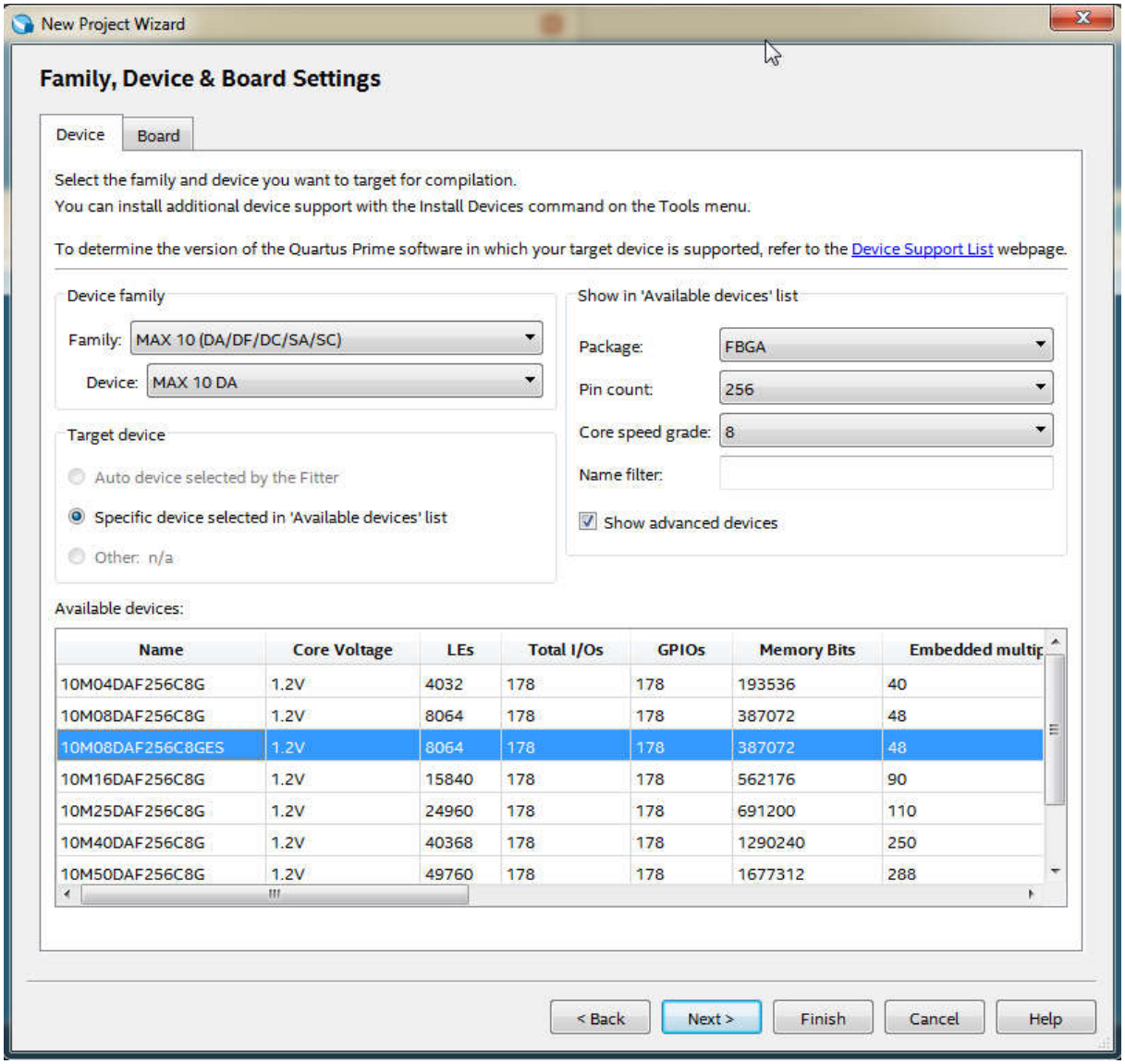
Rys. 3. Wybranie katalogu roboczego, nazwy projektu i modułu top-level.

Po kliknięciu ***Next*** pojawi się okno zapytania czy chcemy utworzyć pusty projekt czy też skorzystać z gotowych wzorców projektów. Wybieramy ***Empty Project*** jak to przedstawiono na rys. 4 poniżej.



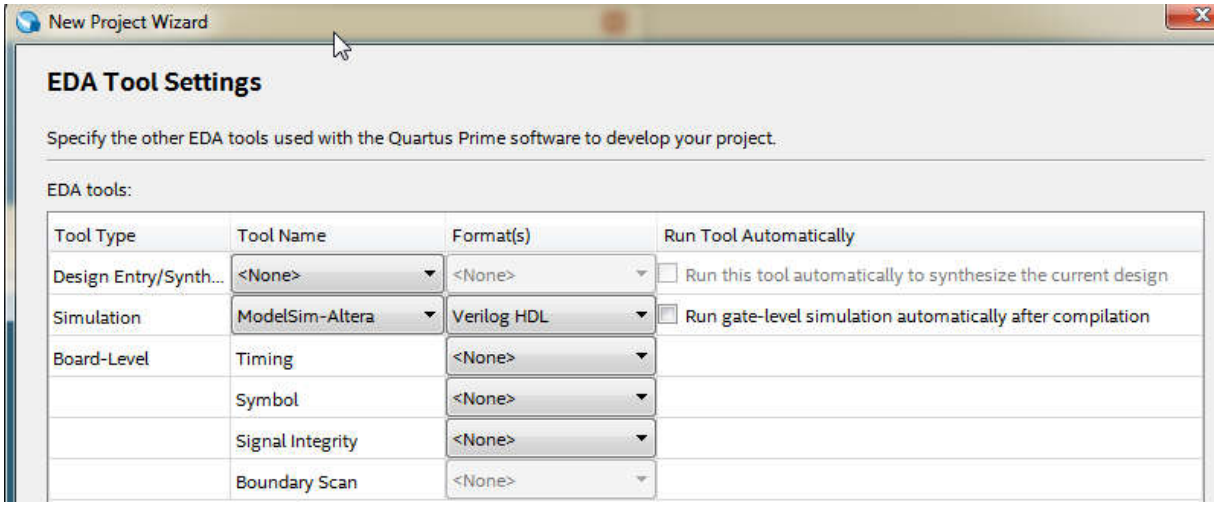
Rys. 4. Wybór pustego projektu.

W kolejnym kroku mamy możliwość dodania gotowych, już istniejących plików do projektu. W naszym przypadku nie dodajemy żadnych plików i klikamy ***Next***. W kolejnym oknie wybieramy układ FPGA, który jest zamontowany na płytce rozwojowej MAXimator, tak jak to przedstawiono na rys. 5.



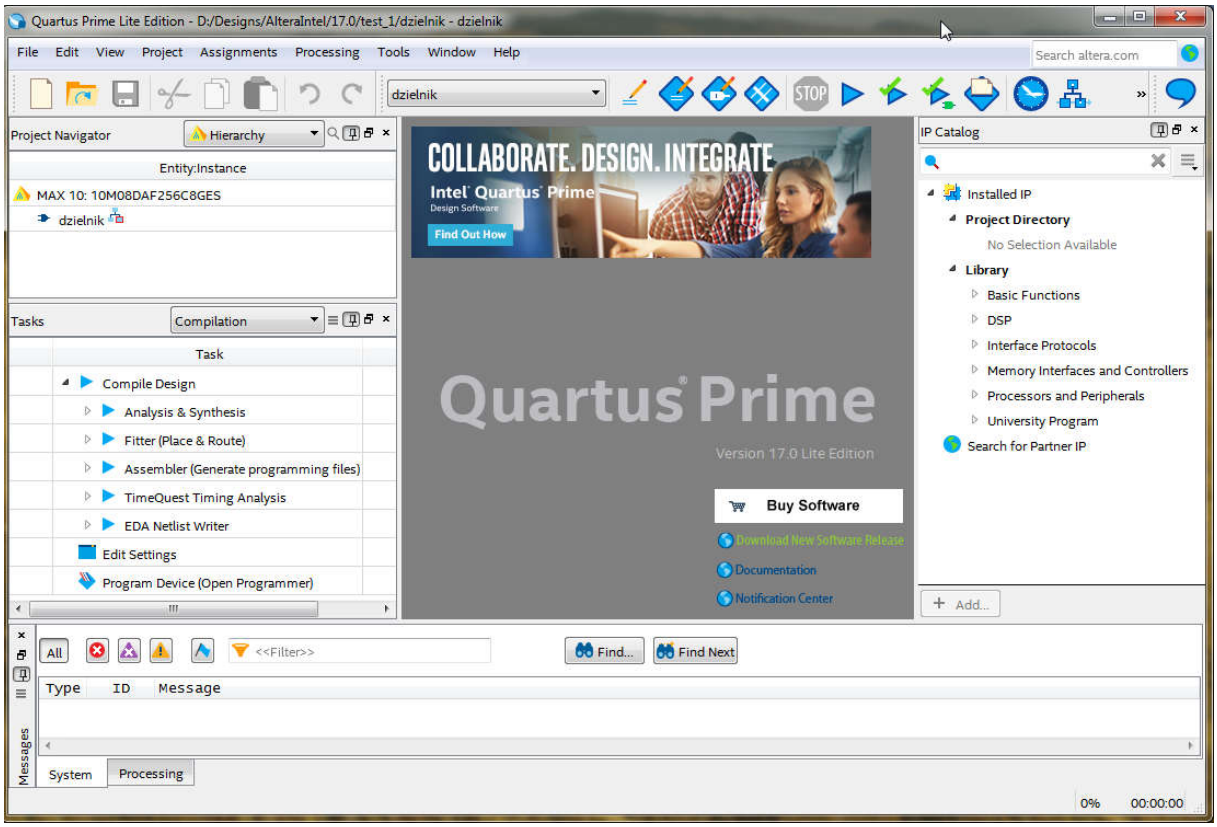
Rys. 5. Wybór układu FPGA typu 10M08DAF256C8GES jako docelowego używanego w projekcie.

W kolejnym kroku wybieramy narzędzia firm trzecich, które są używane w procesie projektowania. W naszym przypadku należy wybrać zewnętrzny symulator ***ModelSim-Altera*** wg poniższego rysunku.



Rys. 6. Wybór symulatora ModelSim-Altera wykorzystywanego do symulacji projektu.

W kolejnym kroku pojawia podsumowanie projektu, klikamy ***Finish***, powinno pojawić się okno programu jak to przedstawiono na rys. 7.



Rys. 7. Widok okna głównego Quatrus Prime po skonfigurowaniu nowego, pustego projektu.

Dodanie pliku Verilog (lub VHDL)

W celu dodania nowego pliku wybieramy menu ***File → New → Verilog HDL File (lub VHDL file)***. Otworzy się nowy pusty plik. Pierwszą czynnością jest zapisanie go pod własną

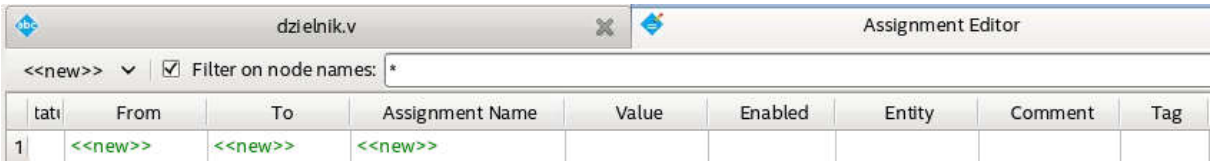
nazwą, wybieramy menu ***File → Save As*** i zapisujemy pod nazwą ***dzielnik.v***. Następnie należy wprowadzić kod w języku Verilog opisujący projektowany układ logiczny. Ponieważ wcześniej zdeklarowano nazwę modułu top-level jako ***dzielnik*** teraz należy ją **koniecznie** wykorzystać. Poniżej jest zamieszczony przykład opisu w języku Verilog prostego dzielnika częstotliwości przez 5 000 000.

```
module dzielnik(clk_i, rst_i, clk_o);
    //  I/O ports
    input clk_i, rst_i;
    output reg clk_o;

    // internal signal
    integer counter;
    // divide factor
    parameter DivRatio = 5_000_000;

    // module behavioral description
    always @(posedge clk_i or posedge rst_i)
    if (rst_i==1)
          begin
               counter = 0;
               clk_o = 0;
          end
    else
          begin
               if (counter==(DivRatio-1))
                    counter = 0;
               else
                    counter = counter + 1;
               if (counter==(DivRatio/2))
                    clk_o = 1;
               if (counter==0)
                    clk_o = 0;
          end
endmodule
```

Po wprowadzeniu opisu i zapisaniu (klawisze ***Ctrl-S*** lub menu ***File → Save***) należy skompilować plik w celu sprawdzenia jego poprawności. W tym celu klikamy na przycisk  (***Analyze Current File***) i sprawdzamy komunikaty w dolnym oknie o nazwie ***Message***. W przypadku pojawienia się błędu, kliknięcie na jego opis automatycznie przenosi kursor w kod Verilog w okolice wykrycia nieprawidłowości. Błędy należy poprawiać zawsze zaczynając od pierwszego ujawnionego w oknie ***Message***!!!

Oprócz plików opisujących działanie logiczne projektowanego układu (w formatach Verilog, VHDL lub schematu graficznego) można wprowadzić dodatkowe wymagania takie jak np. wymagania czasowe czy przypisanie nazw logicznych portów do fizycznych wyprowadzeń układu FPGA. Przed przypisaniem wyprowadzeń należy koniecznie wykonać polecenie ***Run Analysis & Synthesis***, (klawisze ***Ctrl+K*** lub przycisk  lub menu ***Procesing → Start → Start Analysis & Synthesis***), w przeciwnym razie wyprowadzenia

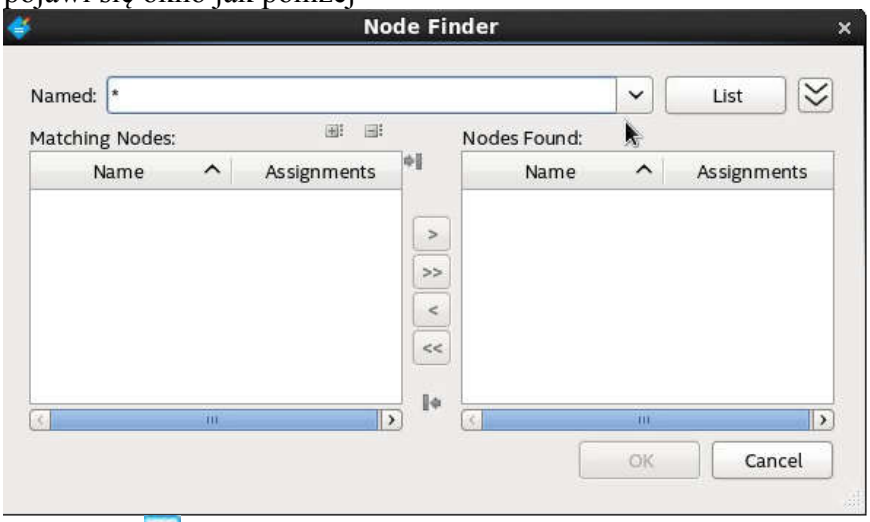
I/O nie będą widoczne. Najprostszą ale czasochłonną metodą przypisania wyprowadzeń jest wybranie menu ***Assignments → Assingment Editor***, pojawi się okno jak na rys. 8.



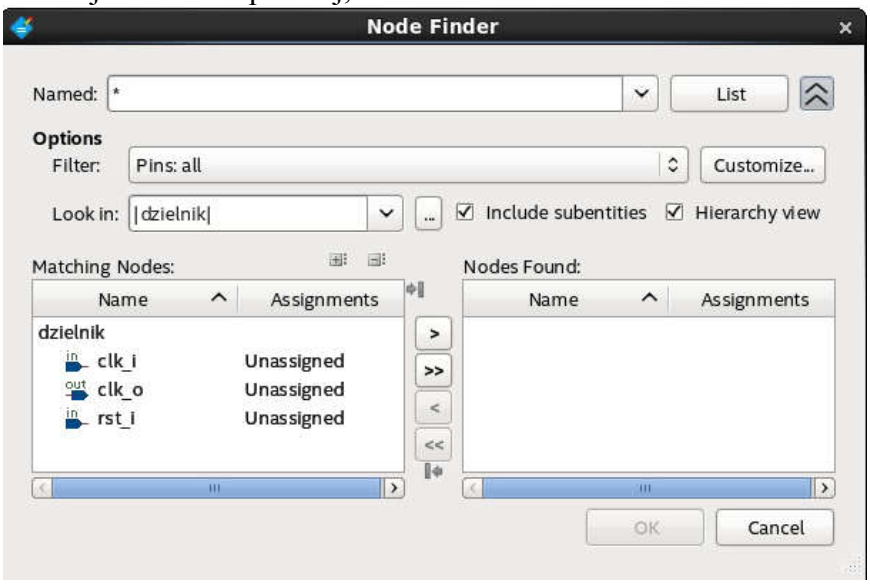
Rys. 8. Okno Assignment Editor bez przypisanych wyprowadzeń układu scalonego do nazw logicznych projektowanego bloku cyfrowego.

W celu przyporządkowania wyprowadzeń należy kolejno wykonać:

- wybrać ***Category - All***,
- jeśli nie ma pustej linii (na rys. 8 jest) kliknąć na napis ***<<new>>*** znajdujący się po lewej górnej stronie,
- podwójnie kliknąć na pole pod pozycją ***To***, a następnie kliknąć na symbol lornetki, pojawi się okno jak poniżej



- kliknąć na  w celu rozwinięcia dostępnych opcji,
- w polu ***Filter*** należy wybrać ***Pins: all*** a następnie kliknąć przycisk ***List***, pojawi się widok jak w oknie poniżej,



- należy kliknąć w nazwę wyprowadzenia do przypisania (np. ***clk_i***) a następnie w przycisk ***>*** i ***OK***, nazwa pojawi się w kolumnie ***To***,

- w kolumnie ***Assignment Name*** należy wybrać ***Location (Accepts wildcards/groups)***,
- w kolumnie ***Value*** należy wpisać nr wyprowadzenia które ma być podłączone do uprzednio wybranej nazwy logicznej (np. ***PIN_L3***),
- czynności powyższe należy wykonać dla każdego wyprowadzenia oddzielnie, a następnie należy konfigurację zapisać (klawisze ***Ctrl+S***),
- okno edytora przypisań powinno wyglądać jak np. poniżej na rys. 9

Rys. 9. Okno przypisań nazw logicznych projektowanego układu cyfrowego do fizycznych wyprowadzeń układu FPGA. Przykład dla kodu dzielnika częstotliwości podanego powyżej i płytki MAXimator.

Łatwiejszą formą wykonania przypisań jest stworzenie pliku tekstowego (koniecznie z rozszerzeniem nazwy *.qsf) a następnie wczytanie go poprzez menu ***Assingments → Import Assingments***. Przykład zawartości takiego pliku przedstawiono poniżej. Tekst za znakiem # jest komentarzem.

```
#General pin assignment format:
#set_location_assignment PIN_xxx -to port_name

# 10MHz clock generator
set_location_assignment PIN_L3 -to clk_i

# Button RES, red, rigth side (pressed low, default high):
set_location_assignment PIN_B16 -to  rst_i

# LED (low level active)
set_location_assignment PIN_M16 -to clk_o # LED0
```

Kompilacja projektu

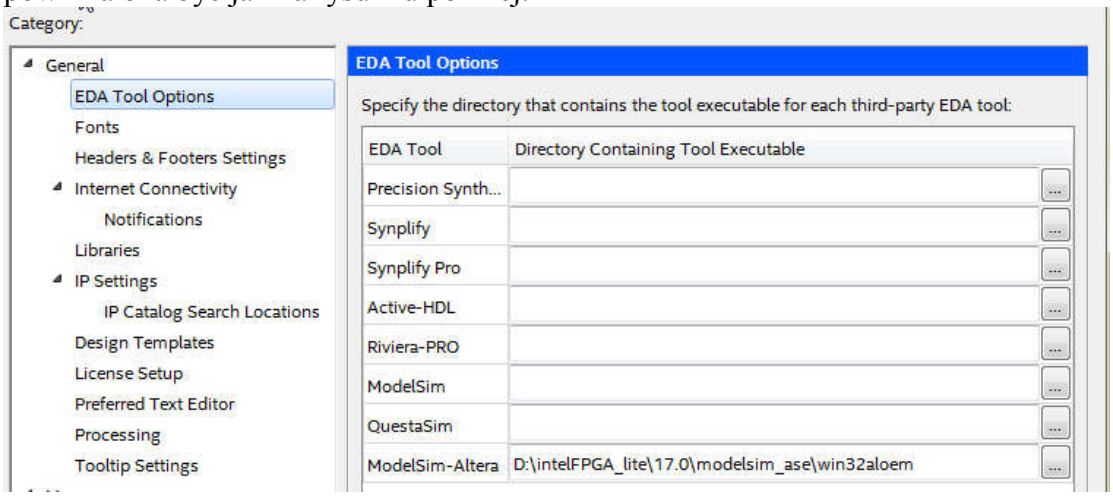
Kompilacja projektu w środowisku Quartus Prime oznacza przygotowanie plików projektowych do wgrania ich na docelową płytkę jak również przygotowanie plików niezbędnych do analizy zaprojektowanego układu w postaci np.: symulacji funkcjonalnej, analizy czasowej, analizy zapełnienia układu itd. Kompilację wykonujemy poprzez przycisk ▶ lub kombinację klawiszy ***Ctrl+L***. Kombinacja klawiszy ***Alt+4*** włącza wyświetlanie okna postępu kompilacji.

Wykonanie symulacji funkcjonalnej

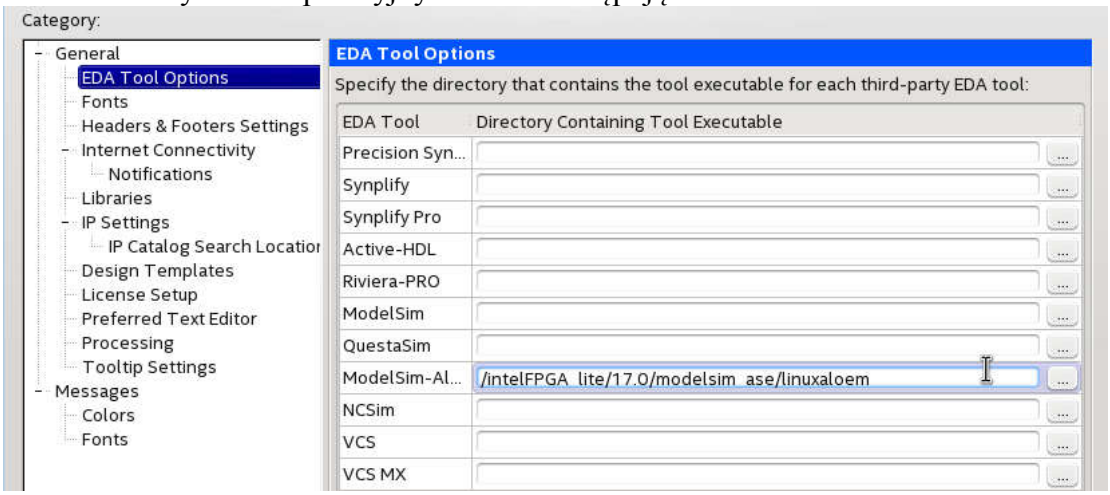
Wykonanie symulacji funkcjonalnej z wykorzystaniem testbenchy w języku Verilog wymaga wykorzystania, dołączanego do pakietu CAD Quartus, dodatkowego oprogramowania symulacyjnego ModelSim Intel FPGA Starter Edition. Jest oprogramowanie firmy trzeciej (Mentor Graphics Co) i do jego użycia niezbędne jest odpowiednie skonfigurowanie środowiska. Przebieg wykonania konfiguracji i symulacji jest następujący:

- skonfigurowanie pod menu ***Tools → Options → Eda Tool Options*** ścieżki dostępu do oprogramowania ModelSim-Altera, poniższe przykłady konfiguracji są dla domyślnej ścieżki instalacji oprogramowania ModelSim, w systemie Windows

powinna ona być jak na rysunku poniżej:



natomiast w systemie operacyjnym Linux następująco:

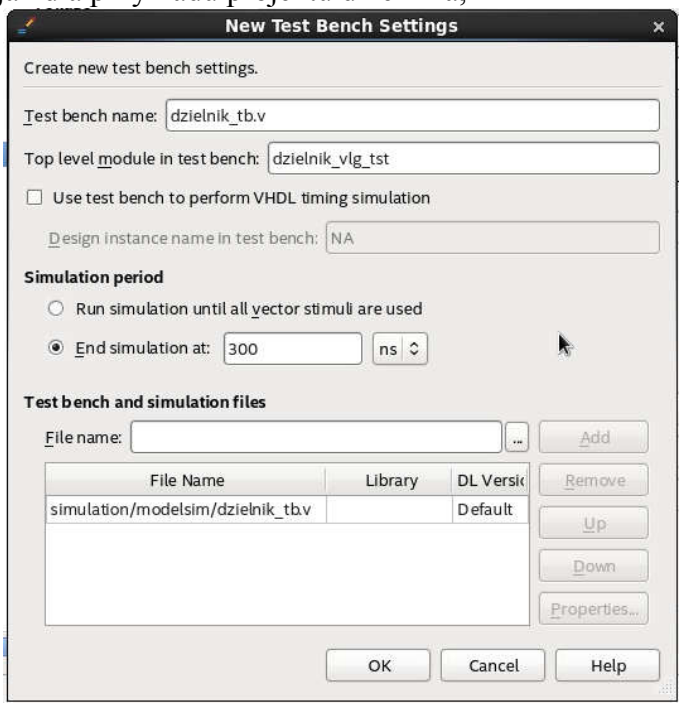


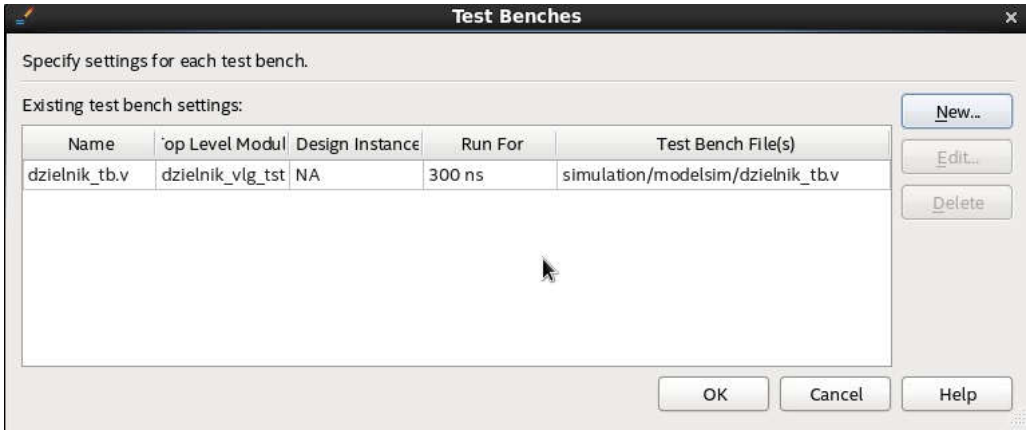
- wygenerowanie szablonu pliku testbench poprzez menu ***Processing → Start → Start Test Bench Template Writer***, w katalogu ***./simulation/modelsim*** pojawi się plik z rozszerzeniem *.vt (dla Veriloga) lub *.vht (dla VHDLa),
- plik wygenerowany w poprzednim kroku należy otworzyć (***Ctrl-O***, należy wybrać rozszerzenie do otwierania typu ***Test Bench Output Files***),
- otworzony wcześniej plik należy zapisać z rozszerzeniem *.v (lub *.vht dla VHDLa), menu ***File → Save As...*** (może być pod inną bardziej przystępną nazwą niż wygenerowana automatycznie), następnie należy ten plik uzupełnić dodając sygnały pobudzenia i inne niezbędne w danym projekcie elementy, należy zanotować nazwę top_level testbencha, poniżej przykład takiego pliku dla projektu ***dzielnik*** wprowadzonego wcześniej:

```
`timescale 1 ns/ 1 ps
module dzielnik_vlg_tst();
   // test vector input registers
   reg clk_i;
   reg rst_i;
   // wires
   wire clk_o;
   // assign statements (if any)
   defparam i1.DivRatio=5;
dzielnik i1 (
```

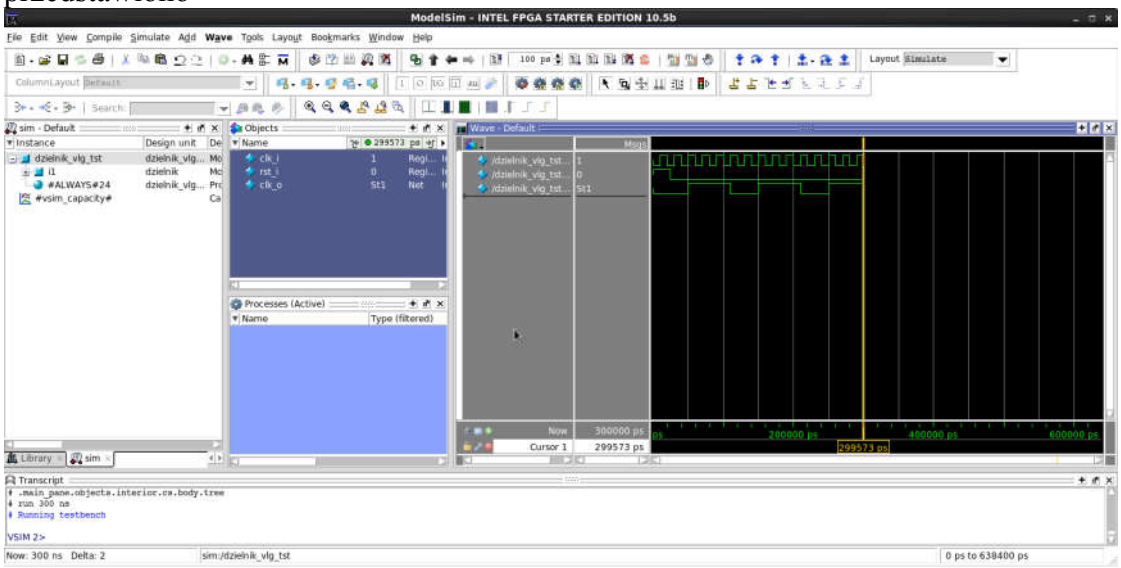
```
// port map - connection between master ports and
signals/registers
    .clk_i(clk_i),
    .clk_o(clk_o),
    .rst_i(rst_i)
);
initial
begin
    $display("Running testbench");
    clk_i=0;
    rst_i=1;
    # 25 rst_i =0;
end

always
    #10 clk_i = ~clk_i;
endmodule
```

- pomyślne skompilowanie projektu (***Ctrl+L***), do symulacji funkcjonalnej wystarczy elaboracja,
- następnie należy podać nazwę pliku testbench i bloku top_level w menu ***Assignments → Settings → EDA Tool Settings → Simulation*** w polu ***NativeLink settings*** należy zaznaczyć pole ***Compile test bench*** a następnie kliknąć pole ***Test Benches*** i dodać (menu ***New...***) utworzony wcześniej plik testbencha podając również nazwę bloku top_level, oraz wpisać czas końca symulacji (inaczej symulacja będzie prowadzona w nieskończoność), poniżej przedstawiono przykład konfiguracji jak dla przykładu projektu dzielnika,





- na koniec uruchamiamy symulację poprzez menu ***Tools → Run Simulation Tool → RTL Simulation***,
- w oddzielnym oknie powinna uruchomić się symulacja jak np. poniżej to przedstawiono



- domyślnie włączane jest bardzo duże przybliżenie wykresu czasowego, aby rozciągnąć wykres na cały czas symulacji należy kliknąć myszką w przebieg czasowy i wcisnąć klawisz ***f***,
- domyślnie na wykresie umieszczane są sygnały zewnętrzne badanego bloku, można na wykresie umieścić każdy inny sygnał dostępny w projekcie, w tym celu należy wybrać moduł i zaznaczyć sygnał a następnie przeciągnąć go myszką na wykres czasowy, aby zaktualizować wartość tego sygnału na wykresie należy zrestartować symulację,
- wszelkie zmiany w pliku testbench wymagają ponownego uruchomienia symulacji, t.j. zamknięcie okna symulacji i ponowne wydanie polecenia menu ***Tools → Run Simulation Tool → RTL Simulation***.

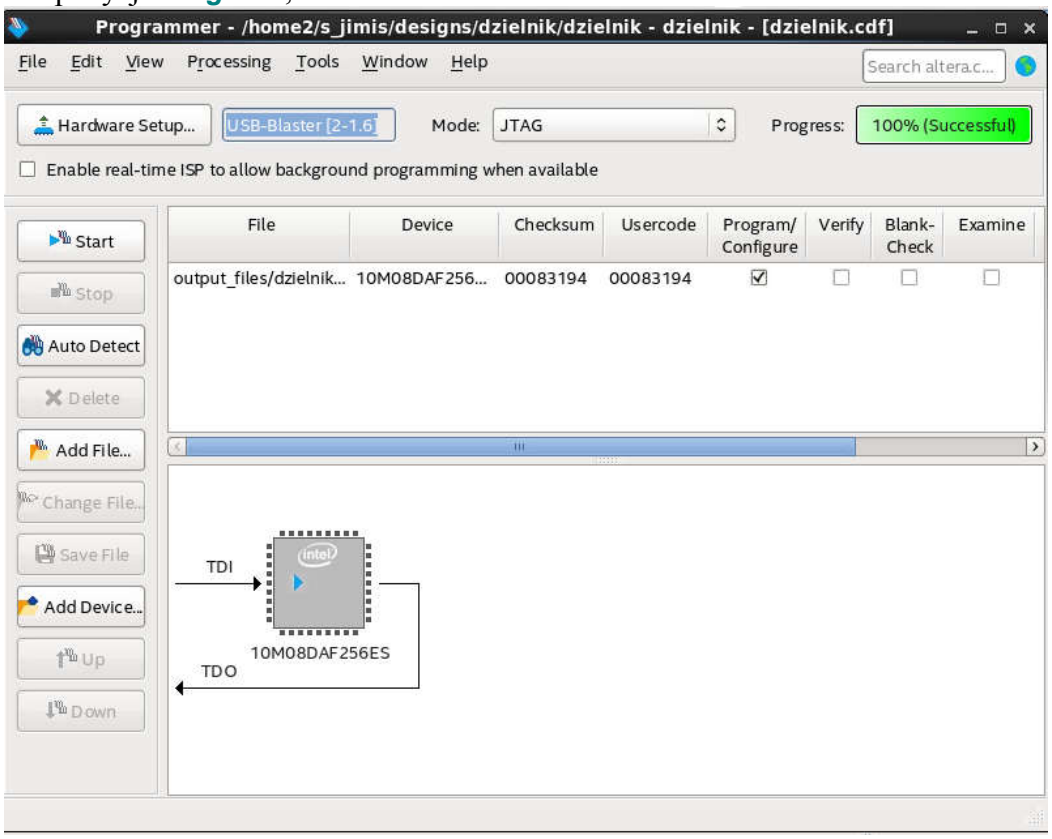
Programowanie układu na płytce uruchomieniowej

Przed programowaniem należy wykonać następujące czynności:

- do płytki MAXimator dołączyć programator USB-Blaster taśmą 10-przewodową,
- do płytki MAXimator podłączyć zasilacz ze złączem micro-USB, alternatywnie można zasilanie podać z komputera poprze kabel micro-USB,
- podłączyć programator USB-Blaster kablem USB do komputera,
- uruchomić programator poprzez menu ***Tools → Programmer***, powinno pojawić się okno jak na poniższym rysunku, w celu ostatecznego zaprogramowania klikamy

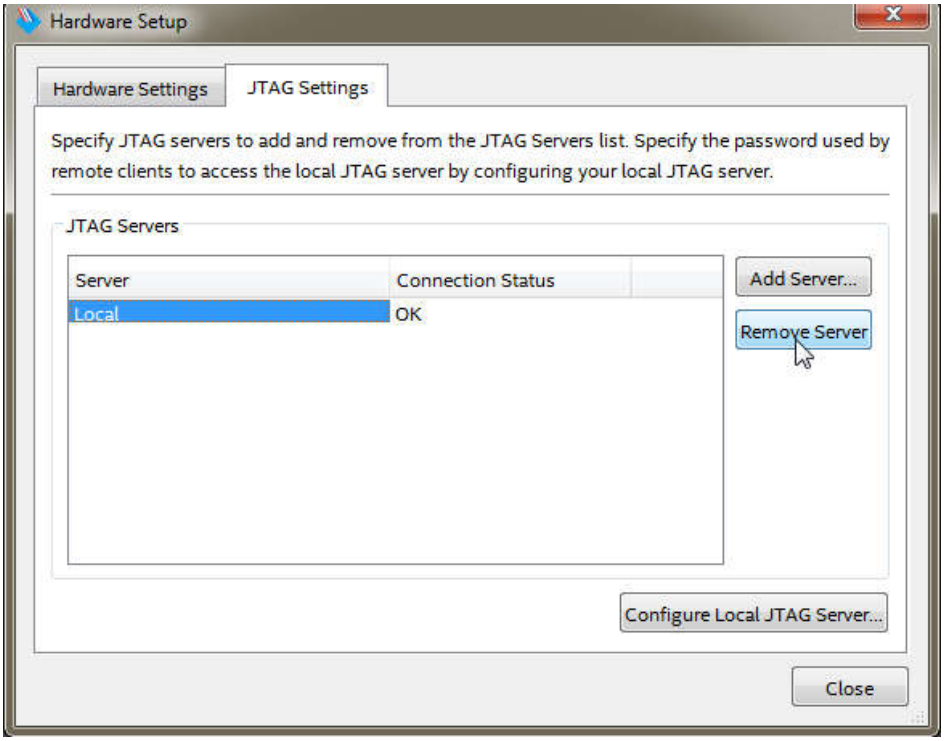
przycisk 

, pomyślne zaprogramowanie sygnalizowane jest zielonym kolorem na pozycji ***Progress***,



- jeśli w oknie jak powyżej przy przycisku ***Hardware Setup...*** jest opis ***No Hardware*** należy wcisnąć ten przycisk i w polu ***Currently selected hardware*** wybrać ***USB — Blaster***.

Czasami w systemie Linux zdarza się, że mimo iż widziany jest i skonfigurowany prawidłowo programator, proces programowania układ MAX10 kończy się błędem. W takich przypadkach należy jeszcze raz wejść w ustawienie ***Hardware Setup*** a następnie w zakładkę ***JTAG Setting***, wybrać ***Local Serwer*** i go usunąć przyciskiem ***Remove Serwer***. Pomimo usunięcia serwer powinien za chwilę ponownie się pojawić a po zamknięciu ustawień ***Hardware Setup*** programowanie powinno przebiegać prawidłowo.



Rys. 10. Usuwanie lokalnego serwera pomaga w problemach programowania płytek w systemie Linux. Po usunięciu wpisu **Local** *powinien się on automatycznie pojawić ponownie. Jeśli to nie nastąpi należy ponownie uruchomić oprogramowanie programatora. Od tego momentu programowanie powinno przebiegać prawidłowo.*

Dodatek 1: Wykonanie symulacji funkcjonalnej w środowisku ModelSim bez użycia oprogramowania Quartus Prime Lite.

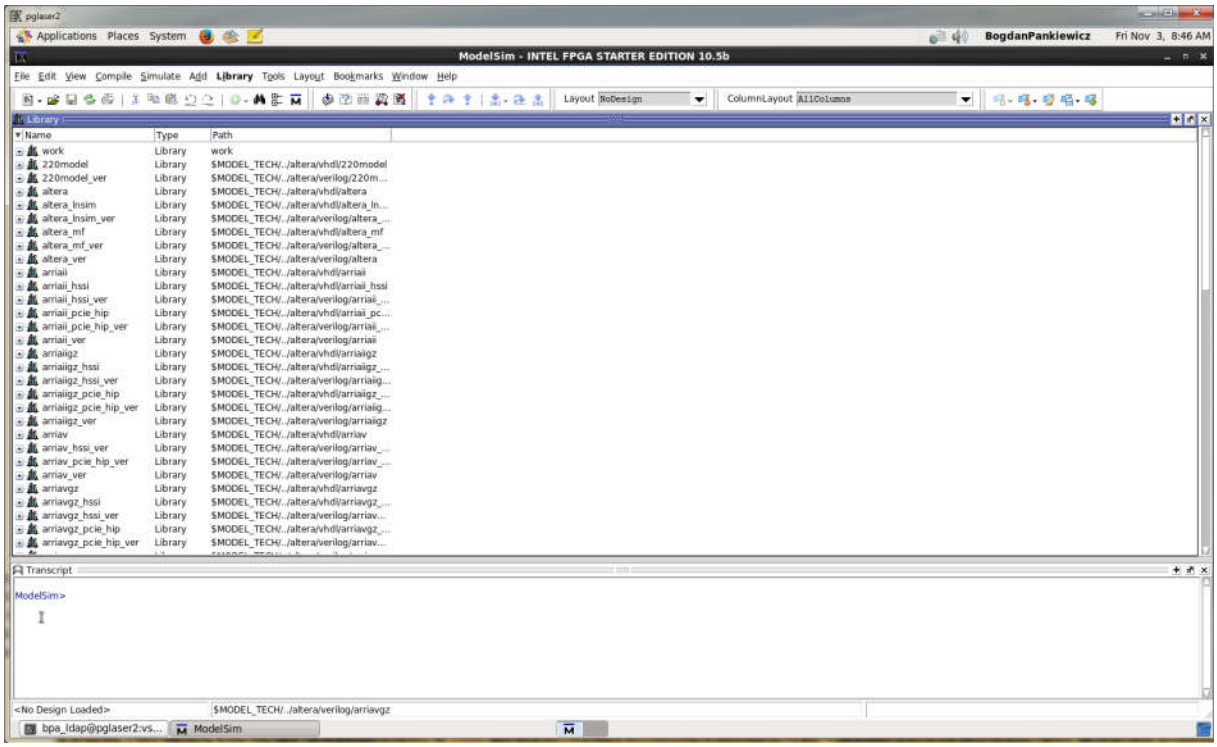
Pierwszą czynnością jest skonfigurowanie środowiska i utworzenie katalogu roboczego identycznie jak dla oprogramowania Quartus Prime Lite. W kolejnym kroku należy skopiować pliki Verilog (opisywanych bloków i testbencha) do katalogu roboczego a następnie należy przejść do tego katalogu i uruchomić oprogramowanie poprzez podanie polecenia ***vsim &***. W celu ewentualnej korekty zawartości plików można użyć dowolnego edytora tekstowego np.: ***gedit &***. Po wydaniu polecenia pojawi się okno oprogramowania jak na rys. 11 poniżej. W kolejnych krokach należy wydawać polecenia tekstowe w oknie ***Transcript*** umieszczonym w dolnej części programu. Są to kolejno następujące polecenia wymienione na liście poniżej:

1) Utworzenie biblioteki roboczej o nazwie `work`:
```
vlib work
```
2) Inicjalizacja biblioteki roboczej i mapowanie z katalogiem roboczym o tej samej nazwie
```
vmap work work
```
3) Wczytanie i skompilowanie plików Verilog (tu można używać tzw. wildcards * i ?):
```
vlog testd.v testd_tb.v
```
W przypadku plików w języku VHDL ich wczytanie wykonuje się poleceniem:
```
vcom -93 testd.vhd testd_tb.vhd
```
W przypadku gdy we wczytywanych plikach są błędy składniowe w oknie ***Transcrypt*** pojawi się stosowny komunikat (czasami należy w niego kliknąć myszką w celu rozwinięcia szczegółów) wskazujący miejsce i prawdopodobną przyczynę błędu. W przypadku wielu błędów poprawki należy wprowadzać w kolejności od pierwszego ujawnionego!

4) W tym momencie należy myszką rozwinąć bibliotekę **work** a następnie kliknąć na blok testbencha. Powinien otworzyć się pusty wykres czasowy i lista sygnałów dostępnych w bloku testbench. Należy zaznaczyć i przeciągnąć na wykres sygnały, które chcemy obserwować.
5) Uruchomienie symulacji na okres np. 200ns:

```
run 200ns
```

6) Domyślnie włączone jest bardzo duże przybliżenie ostatniego fragmentu czasowego symulacji. Aby zmienić powiększenie na pełen zakres czasu symulacji należy myszką kliknąć na wykres czasowy i nacisnąć klawisz ***f***.



Rys. 11. Okno symulatora ModeSim bez załadowanego projektu.

Dodatek 2: Symulacja funkcjonalna w środowisku ModelSim z użyciem skryptu oraz wykorzystaniem Quartus Prime jako edytora tekstowego.

Konfigurowanie i manipulacja na plikach w celu symulacji funkcjonalnej w środowisku ***Quartus Prime*** jest nieco uciążliwa stąd warto zmienić przebieg procesu projektowania i symulację funkcjonalną wykonać niezależnie i samodzielnie wykorzystując oprogramowanie ***ModelSim*** jako symulator wywoływany niezależnie. Dodatkowo w celu przyspieszenia pracy warto wykorzystać proste skrypty symulacyjne. Ponieważ końcowym efektem ma być zaprogramowany układ FPGA warto w początkowej fazie użyć oprogramowanie CAD do stworzenia/edycji plików Verilog i skryptu symulacyjnego. Przebieg projektowania spełniający powyższe założenia zostanie zaprezentowany na przykładzie prostego projektu trzybitowego licznika w kodzie Graya. Kolejność niezbędnych operacji jest przedstawiona poniżej.
1) **Uruchomienie oprogramowania *Quartus Prime* i założenie nowego pustego projektu.** Tą czynność należy wykonać identycznie jak to przedstawiono w części: ***Utworzenie nowego projektu***. Dla celów niniejszego przykładu projekt nazywamy ***gray***.

2) **Dodanie pliku Verilog**. Następnie, podobnie jak poprzednio, należy dodać plik Verilog o nazwie ***gray.v*** i zawartości np. jak przedstawiona poniżej.

```
module gray(clk_i, rst_ni,gray_o);
   input clk_i, rst_ni;
   output reg [3:0] gray_o;

   always @(posedge clk_i or negedge rst_ni)
      if (!rst_ni)
         gray_o=3'b000;
      else
         case (gray_o)
            3'b000: gray_o=3'b001;
            3'b001: gray_o=3'b011;
            3'b011: gray_o=3'b010;
            3'b010: gray_o=3'b110;
            3'b110: gray_o=3'b111;
            3'b111: gray_o=3'b101;
            3'b101: gray_o=3'b100;
            3'b100: gray_o=3'b000;
            //default: gray_o=0;
         endcase
endmodule
```

3) **Dodanie pliku testbench**. Następnie należy utworzyć plik Verilog testbencha o nazwie np. ***gray_tb.v*** i zawartości np. jak przedstawiona poniżej.

```
`timescale 1ns/1ps
module gray_tb();

   reg clk_i,rst_ni;
   wire [2:0] gray_o;

   gray g1(clk_i, rst_ni, gray_o);

   initial
      begin
         $monitor("Input signals: clk_i=%b, rst_ni=%b, gray_o=%b,
time=%t", clk_i, rst_ni, gray_o, $time);
         clk_i=0;
         rst_ni=0;
         #20 rst_ni=1;
      end
   always
      #10 clk_i=!clk_i;
endmodule
```

4) **Utworzenie skrytpu symulacyjnego**. Wybieramy menu ***File/New/Text File*** i zapisujemy plik pod nazwą np. ***do.do***. Przykładowa zawartość pliku przedstawiona jest poniżej.

```
transcript on
if {[file exists work]} {
   vdel -lib work -all
```

```
}

vlib work
vmap work work
vlog gray.v
vlog gray_tb.v
vsim gray_tb
add wave *
view structure
view signals
run 200ns
```

5) **Uruchomienie symulacji.** W oknie terminala uruchamiamy symulację wykonując polecenie:

```
vsim -do do.do &
```

W przypadku wykrycia błędów w oknie ***Transcrypt*** pojawi się komunikat o pliku i linii w pliku w którym wykryto błąd. Należy ten błąd poprawić (poprzez edycję w ***Quartus Prime***) i ponownie uruchomić symulację. Restart symulacji można wykonać bez wychodzenia z oprogramowania ModelSim poprzez wpisanie w jego konsoli polecenia:

```
do do.do
```

6) **Zakończenie projektu**. Po potwierdzeniu poprzez symulację poprawności projektu należy dodać przypisania wyprowadzeń (zaimportować plik ****.qsf***), skompilować (***Ctrl+L***) i wgrać do układu FPGA tak jak to opisano w głównej części niniejszej instrukcji.

Dodatek 3: Niektóre polecenia systemu Linux (shell tcsh).

A) Informacje wstępne

Polecenia w terminalu systemu ***Linux*** podaje się poprzez wprowadzenie nazwy polecenia i naciśnięcie klawisza ***Enter***. Możliwe jest również uruchomienie polecenia w tle - aby to wykonać polecenie należy zakończyć znakiem ***&*** (z przerwą lub bez przerwy między poleceniem za znakiem ***&***). Uruchomienie w tle spowoduje „zwolnienie" terminala i można w nim wykonywać kolejne polecenia. Przykład uruchomienia graficznego edytora teksu w nowym oknie i zwolnienie terminala do innych prac:

```
gedit &
```

W przypadku poleceń wykonywanych natychmiastowo dodawanie znaku ***&*** nie ma sensu bo terminal i tak zostaje zwolniony po zakończeniu wykonywania polecenia, nie jest to jednak żadnym błędem.

> ***Uwaga:*** *system Linux jest czuły na wielkość liter zarówno jeśli chodzi o wprowadzane polecenia jak i nazwy plików!*

W przypadku używania shella tcsh możliwe są pewne wbudowane udogodnienia:

- naciśniecie klawisza ***TAB*** w czasie wprowadzania nazw plików i katalogów wykonuje funkcję autozakańczania nazw plików, jeśli w danym katalogu jest kilka możliwości zakończenia nazwy są one wylistowane w celu pomocy wyboru operatorowi, np. napisanie w terminalu litery ***a*** i naciśnięcie klawisza ***TAB*** wylistuje wszystkie pliki zaczynające się literą ***a*** lub jeśli będzie tylko jeden taki plik spowoduje wypisanie jego nazwy,
- naciskanie strzałek góra/dół przywołuje ostatnio wykonane polecenia i można je łatwo powtórzyć bez konieczności ponownego ich wpisywania,
- zaznaczenie teksu występującego w terminalu poprzez użycie lewego klawisza myszki powoduje automatyczne skopiowanie tego tekstu do schowka, naciśnięcie

prawym klawiszem myszki w danym miejscu powoduje automatyczne wklejenie tekstu ze schowka.

Rodzaj używanego shell-a można sprawdzić poprzez wydanie polecenia:

```
ps
```

W odpowiedzi uzyskamy wszystkie procesy pracujące w tle. W przypadku świeżo otworzonego terminala wynik może być jak poniżej umieszczony:

```
[s_test@pglaser2 ~]$ ps
  PID TTY          TIME CMD
19519 pts/9    00:00:00 tcsh
19727 pts/9    00:00:00 ps
```

Obecność procesu o nazwie ***tcsh*** świadczy o używaniu właśnie tego shella. Polecenia wydawane w terminalu mają swój system pomocy, który możemy uruchomić poprzez podanie nazwy danego polecenia z parametrem wywołania ***--help***. Przykład wywołania pomocy dla polecenia ***ps*** przedstawiony jest poniżej:

```
ps --help
```

Poniżej umieszczono fragment pomocy uzyskanej z powyższego polecenia:

```
[s_test@pglaser2 ~]$ ps --help
********* simple selection *********  ********* selection by list *********
-A all processes                      -C by command name
-N negate selection                   -G by real group ID (supports names)
...
```

Pomoc dotyczącą systemu można także uzyskać za pomocą polecenia ***man***. Przykład użycia polecenia do odszukania pomocy dotyczącej polecenia ***more***:

```
man more
```

Przewijanie pomocy uzyskuje się klawiszami strzałek góra/dół i spacji a wyjście z polecenia ***man*** uzyskuje się poprzez naciśnięcie klawisza ***”q”***.

B) Operacje na katalogach i plikach.

System plików w systemie Linux nie ma tak zwanych napędów oznaczanych kolejnymi literami a będących zazwyczaj odpowiednikiem fizycznego urządzenia takiego jak np. HDD, CDROM/DVD, FDD lub napęd USB znanego w systemie Windows. Zamiast tego system plików jest zorganizowany jako drzewo zaczynające się od korzenia (oznaczanego znakiem ukośnika ***„/”***) a poszczególne napędy zmapowane są do podkatalogów. Najczęściej główny dysk komputera (jedna z jego partycji lub też zasób złożony z kilku dysków) zmapowana jest do korzenia ***/***. Kolejne dyski, jeśli występują, są zazwyczaj zmapowane do jakiegoś podkatalogu np. do katalogu ***/mnt***. Katalogi domowe użytkowników w systemie z domyślnymi ustawieniami są zazwyczaj umieszczone jako podkatalogi w folderze ***/home***.

> ***Uwaga:*** *na komputerach w sieci KSMI katalogi domowe użytkowników są umieszczone w katalogu* ***/home2***.

Każdy użytkownik ma pełen dostęp do swojego katalogu domowego. Poza tym katalogiem użytkownik ma zazwyczaj dostęp tylko do odczytu do wybranych plików systemowych. Dany użytkownik, będący równocześnie właścicielem swojego katalogu domowego może zmienić prawa dostępu do własnych plików i np. umożliwić dostęp do nich innym użytkownikom. Poniżej przedstawione zostaną podstawowe polecenia służące do pracy na plikach i katalogach. W poniższych przykładach założono, że katalog domowy jest skonfigurowany jako ***/home2*** a do systemu zalogował się użytkownik o loginie ***s_test***.

Wyświetlenie bieżącego katalogu:

```
pwd
```

Zmiana katalogu na katalog domowy:

```
cd
```

w wyniku plecenia katalog zostanie zmieniony na ***/home2/s_test***.

Zmiana katalogu na katalog ***/etc***, jest to katalog systemowy do którego użytkownicy mają zazwyczaj dostęp tylko do odczytu

```
cd /etc
```

Wyświetlenie zawartości katalogu bieżącego:

```
ls
```

w wyniku wykonania powyższego polecenia w terminalu są wylistowane nazwy plików i katalogów umieszczonych w bieżącym katalogu.
Wyświetlenie zawartości katalogu bieżącego wraz ze szczegółowymi informacjami i plikami ukrytymi:

```
ls -la
```

w wyniku wykonania powyższego polecenia w terminalu są wylistowane nazwy plików wraz z: prawami dostępu, właścicielem, grupą, wielkością i datą ostatniej modyfikacji.
Symbol kropki ***"."*** oznacza w Linuxie katalog bieżący a dwie kropki ***".."*** oznaczają katalog nadrzędny, stąd zmiana o jeden katalog wyżej będzie możliwa poprzez podanie polecenia:

```
cd ..
```

a wyświetlenie zawartości bieżącego katalogu może być wykonane poprzez:

```
ls
```

lub równoznacznie poprzez:

```
ls .
```

lub też poprzez

```
ls ./
```

Tworzenie nowego katalogu (będącego podkatalogiem w bieżącym katalogu) wykonuje się poprzez:

```
mkdir nazwa_nowego_katalogu
```

Kopiowanie plików wykonuje się poprzez polecenie ***cp***, można przy tym stosować znak gwiazdki ***"*"*** oznaczający dowolny ciąg znaków i znak zapytania ***"?"*** oznaczający dowolny pojedynczy znak. Poniżej zamieszczono kilka przykładów:
Skopiowanie pliku ***a.txt*** z bieżącego katalogu do katalogu o nazwie ***podkatalog***:

```
cp a.txt podkatalog
```

W powyższym przykładzie katalog o nazwie ***podkatalog*** musi istnieć wcześniej niż wydane polecenie kopiowania, w przeciwnym razie plik o nazwie ***a.txt*** zostanie skopiowany do nowego pliku o nazwie ***podkatalog***.
Skopiowanie wszystkich plików których nazwa rozpoczyna się literą ***a*** i kończy ***.txt*** (pomiędzy ***a*** i ***.txt*** może występować dowolny ciąg znaków) do katalogu o nazwie ***podkatalog***:

```
cp a*.txt podkatalog
```

Kopiowanie tworzy nowy plik o identycznej zawartości w miejscu docelowym pozostawiając stary plik w miejscu źródłowym. Jeśli chcemy aby stary plik nie pozostawał w systemie możemy go skasować albo do kopiowania użyć polecenia ***mv***. Polecenie ***mv*** działa identycznie jak ***cp*** nie pozostawiając jednak oryginału w miejscu źródłowym.
Kasowanie plików wykonuje się poprzez polecenie ***rm***. Przykład, skasowanie pliku ***a.txt*** w podkatakogu o nazwie ***podkatalog***:

```
rm ./podkatalog/a.txt
```

Przykład, skasowanie podkatalogu o nazwie ***podkatalog*** wraz z jego całą zawartością (uwaga: polecenie z opcją ***-rf*** należy stosować bardzo ostrożnie bo można łatwo skasować ważne dane):

```
rm -rf podkatalog
```

Zmianę właściciela do pliku/katalogu wykonuje się poprzez wykorzystanie polecenia ***chown*** a zmianę praw dostępu do plików poprzez polecenie ***chmod***.

C) Operacje na zmiennych środowiskowych.

Na wstępie tego podrozdziału należy wyraźnie zaznaczyć, że poniższe polecenia odnoszą się wyłącznie do shella tcsh. Do powołania i ustalenia wartości zmiennej środowiskowej służy polecenie ***setenv***. Zmienne środowiskowe mogą przy tym być powołane do życia bez posiadania jakiejkolwiek wartości.

Powołanie zmiennej środowiskowej o nazwie ***ABC*** bez nadawania wartości:

```
setenv ABC
```

Usunięcie zmiennej środowiskowej ***ABC***:

```
unset ABC
```

Powołanie zmiennej środowiskowej o nazwie ***ABC*** z nadaniem wartości równej ***value_abc***:

```
setenv ABC value_abc
```

Odczyt wartości zmiennej środowiskowej dokonywany jest poprzez podanie znaku dolara ***$*** przed nazwą zmiennej i tak dla polecenia:

```
echo $ABC
```

powinniśmy uzyskać następujący wynik:

```
[s_test@pglaser2 ~]$ echo $ABC
value_abc
```

Niektóre zmienne środowiskowe mają szczególne znaczenie i ich prawidłowe skonfigurowanie jest bardzo ważne dla poprawnej pracy sytemu. Polecenie ***setenv*** bez podania parametrów listuje wszystkie aktualnie istniejące zmienne środowiskowe wraz z ich wartościami:

```
setenv
```

Skrócony wynik powyższej operacji przedstawiony jest tu:

```
[s_testp@pglaser2 ~]$ setenv
USER=s_test
HOME=/home2/s_test
PATH=/usr/lib64/qt-3.3/bin:/usr/local/bin:/bin:/usr/bin
HOSTTYPE=x86_64-linux
VENDOR=unknown
OSTYPE=linux
PWD=/home2/bpa_ldap
GROUP=staff
HOST=pglaser2
REMOTEHOST=bm2.ue.eti.pg.gda.pl
HOSTNAME=pglaser2
LS_COLORS=rs=0:di=01;
CVS_RSH=ssh
G_BROKEN_FILENAMES=1
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
LANG=en_US.UTF-8
ABC=value_abc
```

Jak widać powyżej, w zmiennych środowiskowych zapisywanych jest wiele istotnych dla bieżącej pracy parametrów. Zmienna ***PATH*** wyszczególnia wszystkie katalogi w których będzie wyszukiwane polecenie wpisane w terminalu i zlecone do wykonania poprzez naciśnięcie klawisza ***Enter***.

D) Pozostałe przydatne programy i polecenia.

Zmiana hasła dostępu do konta:

```
passwd
```

Uruchomienie graficznego edytora tekstowego ***gedit*** w tle wraz z utworzeniem pliku ***a.txt*** w bieżącym katalogu:

```
gedit a.txt &
```

Wyświetlenie zawartości pliku tekstowego ***a.txt*** w terminalu:

```
more a.txt
```

lub

```
less a.txt
```

lub

```
cut a.txt
```

Uruchomienie programu analizującego wykorzystanie zasobów (wyjście z programu – klawisz ***q***):

```
top
```

Znalezienie miejsca z którego wykonywane jest dane polecenie: ***which***. Przykład poszukiwania ścieżki do pliku wykonywalnego edytora ***gedit***:

```
which gedit
```

daje wynik w postaci:

```
[s_test@pglaser2 ~]$ which gedit
/usr/bin/gedit
```

Poszukiwania plików dokonuje się przy użyciu polecenia ***find***. Składnia polecenia jest dość złożona więc przedstawiony zostanie wyłącznie jeden przykład polegający na próbie odnalezienia pliku o nazwie ***fstab*** w katalogu ***/etc***:

```
find /etc -name "fstab"
```

Wyniki wyszukiwania przedstawione są poniżej i przedstawiają próby wejścia do każdego z podkatalogów folderu ***/etc*** wraz z niepowodzeniami (bo dostęp jest zabroniony przez system). Odnaleziony został jeden plik w położeniu ***/etc/fstab***.

```
[s_test@pglaser2 ~]$ find /etc -name "fstab"
find: `/etc/ntp/crypto': Permission denied
find: `/etc/audisp': Permission denied
find: `/etc/pki/rsyslog': Permission denied
find: `/etc/pki/CA/private': Permission denied
find: `/etc/cups/ssl': Permission denied
find: `/etc/sssd': Permission denied
/etc/fstab
find: `/etc/dhcp': Permission denied
find: `/etc/selinux/targeted/modules/active': Permission denied
find: `/etc/audit': Permission denied
find: `/etc/sudoers.d': Permission denied
find: `/etc/polkit-1/localauthority': Permission denied
find: `/etc/lvm/cache': Permission denied
find: `/etc/lvm/backup': Permission denied
find: `/etc/lvm/archive': Permission denied
```

Uruchomienie managera plików ***Midnight Commander***:

```
mc
```

Uruchomienie w tle przeglądarki internetowej ***Mozilla Firefox***:

```
firefox &
```

Uruchomienie w tle przeglądarki internetowej ***Mozilla Firefox*** wraz z jednoczesnym wczytaniem wszystkich plików typu ***pdf*** z katalogu bieżącego:

```
firefox *.pdf &
```

Spakowanie całego katalogu o nazwie ***podkatalog*** do jednego pliku o nazwie ***podkatalog.tar.gz*** (poleceniem ***tar***) i jego skompresowanie poleceniem ***gzip***:

```
tar -zcvf podkatalog.tar.gz podkatalog
```

Rozpakowanie spakowanego i skompresowanego archiwum o nazwie ***podkatalog.tar.gz*** do bieżącego katalogu:

```
tar -zxvf podkatalog.tar.gz
```

lub do dowolnego innego katalogu:

```
tar -zxvf podkatalog.tar.gz -C /home2/s_test/katalog_docelowy/
```

Zdalne kopiowanie plików pomiędzy komputerami z systemem Linux pracującymi w sieci możliwe jest poprzez polecenie ***scp***. W poniższym przykładzie użytkownik ***s_test*** (musi on posiadać konta na obu komputerach ***fpgalab12*** i ***pglaser2)*** wykonuje kopię katalogu ***folder_zrodlowy*** wraz ze wszystkimi plikami i podkatalogami z komputera ***fpgalab12*** na komputer ***pglaser2*** do katalogu ***folder_docelowy***. Zarówno ***folder_zrodlowy*** jak i ***folder_docelowy*** umieszczone są w katalogach domowych a użytkownik ***s_test*** jest obecnie zalogowany do komputera ***fpgalab12***:

```
scp -r folder_zrodlowy s_test@pglaser2:/home2/s_test/folder_docelowy
```

Sprawdzenie wykorzystania systemu dyskowego:

```
df -h
```

Sprawdzenie zajętości bieżącego katalogu (wraz z podkatalogami):

```
du –sh .
```

Sprawdzenie zajętości z wyszczególnieniem zajętości dla poszczególnych podkatalogów:

```
dh –h .
```

Literatura:

1. Quartus Prime Introduction Using Verilog Designs", Intel Corporation - FPGA University Program, November 2016, https://www.altera.com/support/training/university/materials-tutorials.html.

Przygotował: Bogdan Pankiewicz, listopad 2017r.