Technikum Łączności ***im. Obrońców Poczty Polskiej w Gdańsku***

Pracownia elektryczna i elektroniczna

(laboratorium techniki cyfrowej)

Projektowanie układów programowalnych FPGA

w środowisku QUARTUS II

(z wykorzystaniem edytora graficznego)

Opracowała mgr inż. Irena Hoja

Gdańsk 2008r.

Spis treści

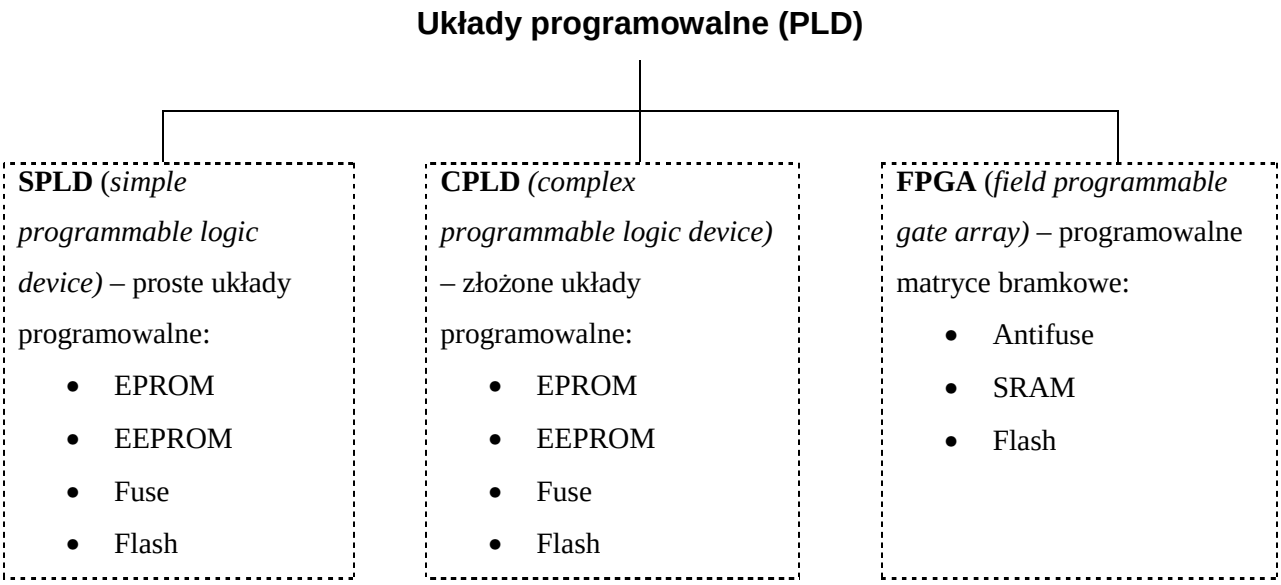
I. Układy programowalne – wprowadzenie

Programowalne układy logiczne (**PLD** – *programmable logic device*) są to układy scalone, których właściwości funkcjonalne są definiowane przez użytkownika. Układy te programuje się w laboratorium używając odpowiedniego oprogramowania narzędziowego i programatorów lub bezpośrednio na płytce docelowej. Kryterium podziału układów programowalnych są cechy ich architektury czyli budowy logicznej (liczby, rodzaju i rozłożenia elementów logicznych i połączeń między nimi).

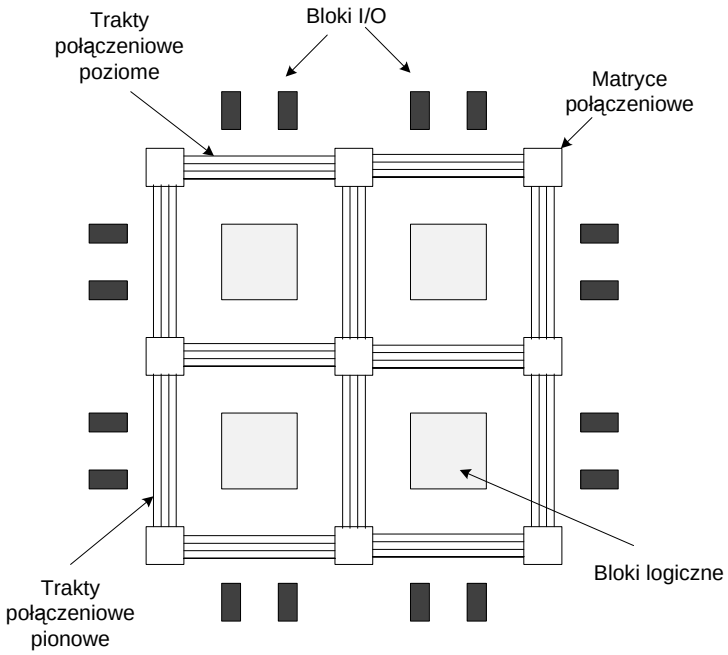
Pomysł konfigurowania układów narodził się wraz z wynalezieniem pamięci półprzewodnikowych, które zawierały dwa podstawowe bloki:

- matryce komórek pamiętających i
- dekoder adresów umożliwiający wybranie (zaadresowanie) grupy komórek pamiętających kombinacje sygnałów na wejściach adresowych pamięci.

Klasyfikacja układów programowalnych:



W układach FPGA poszczególne bloki są łączone ze sobą za pośrednictwem linii traktów połączeniowych oraz programowalnych matryc elementów połączeniowych (kluczy tranzystorowych lub elementów *antifuse*) umieszczonych w miejscach krzyżowania się traktów poziomych i pionowych *(rys.1).*

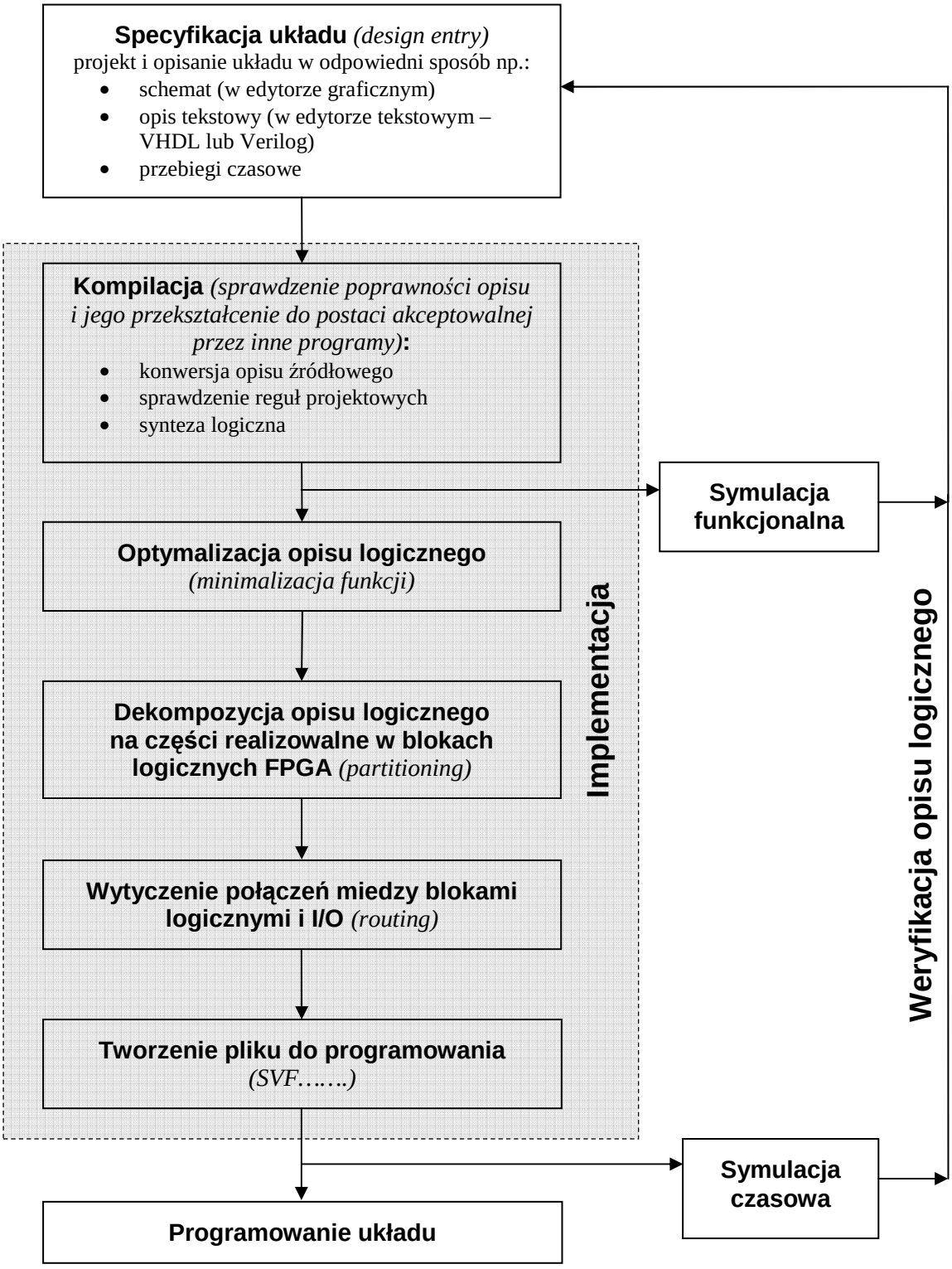


Rys. 1. *Szkic architektury układów FPGA*

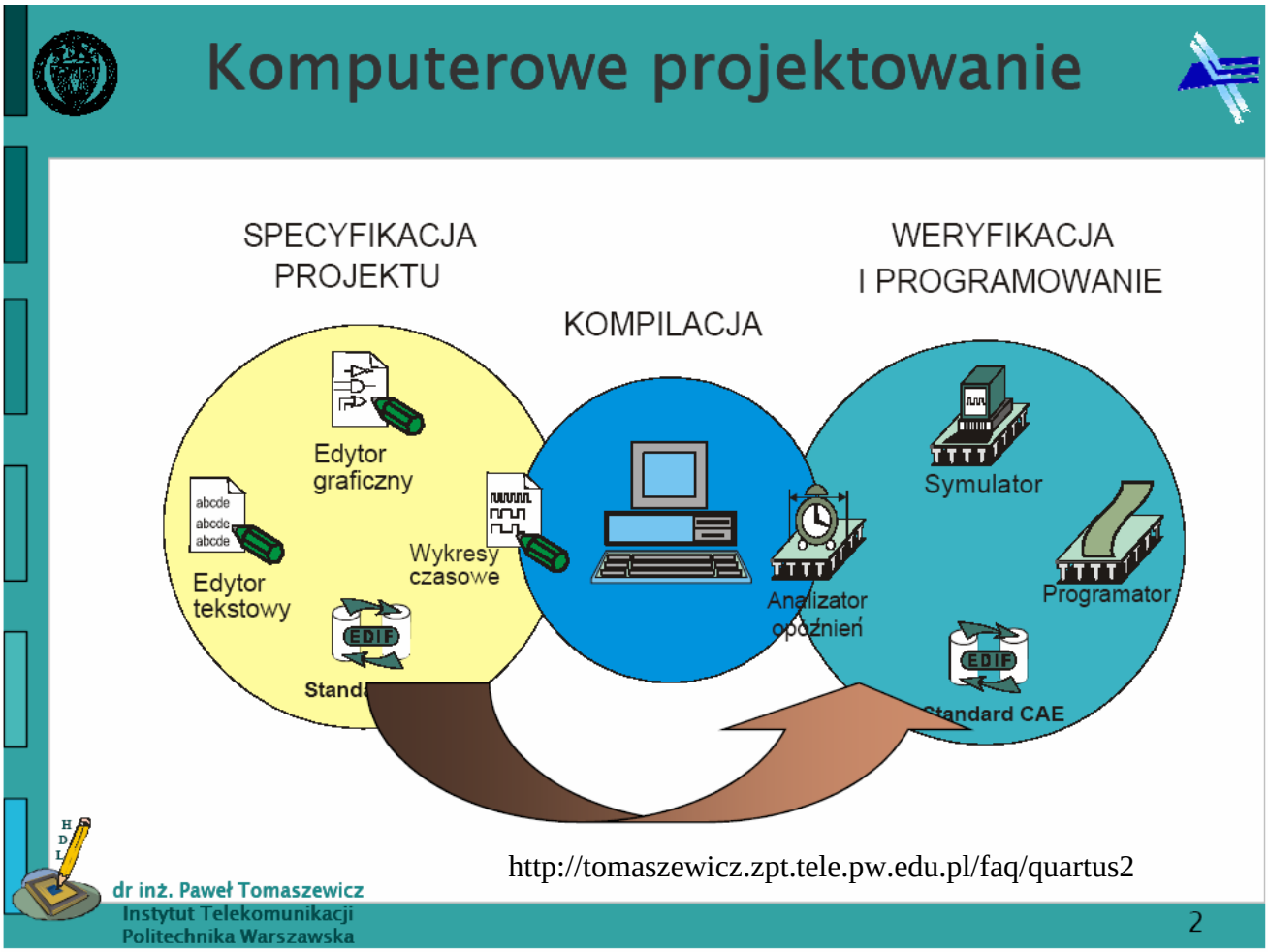
Na obrzeżach matrycy bloków logicznych znajdują się programowalne bloki I/O (wej. – wyj.). Struktury FPGA zawierają od 64 do dziesiątków tysięcy bloków logicznych o bardzo zróżnicowanej budowie. Zazwyczaj złożone bloki logiczne zawierają dwie lub więcej pamięci SRAM umożliwiających generowanie funkcji przełączających za pomocą tablic wartości funkcji LUT *(look – up table)*. W większości układów są to tablice czterowejściowe. W układach o prostej budowie, bloki logiczne zawierają zwykle dwuwejściowe układy generacji funkcji przełączających lub multipleksery czterowejściowe oraz ewentualnie przerzutniki.

Układy FPGA są konfigurowane zazwyczaj za pomocą tranzystorów MOS dołączonych do wejść komórek pamięci statycznych RAM lub za pomocą łączników zwarciowych *antifuse*. W układach FPGA ścieżki połączeniowe (pogrupowane w traktach poziomych i pionowych) są w większości podzielone na segmenty o różnej długości, z których poprzez łączenie poszczególnych fragmentów np. kluczami tranzystorowymi lub łącznikami *antifuse* jest zestawiana wymagana linia połączeniowa. Do realizacji takiego połączenia potrzebnych jest wiele łączników. Powoduje to wzrost rezystancji połączenia i pojemności ścieżki połączeniowej, a w efekcie zwiększenie czasu propagacji sygnału przenoszonego tą ścieżką.

Etapy realizacji układu cyfrowego w strukturze programowalnej PLD przedstawiono na *rys. 2a i rys.2b.*



Rys. 2a. *Etapy realizacji układu w strukturze programowalnej PLD*

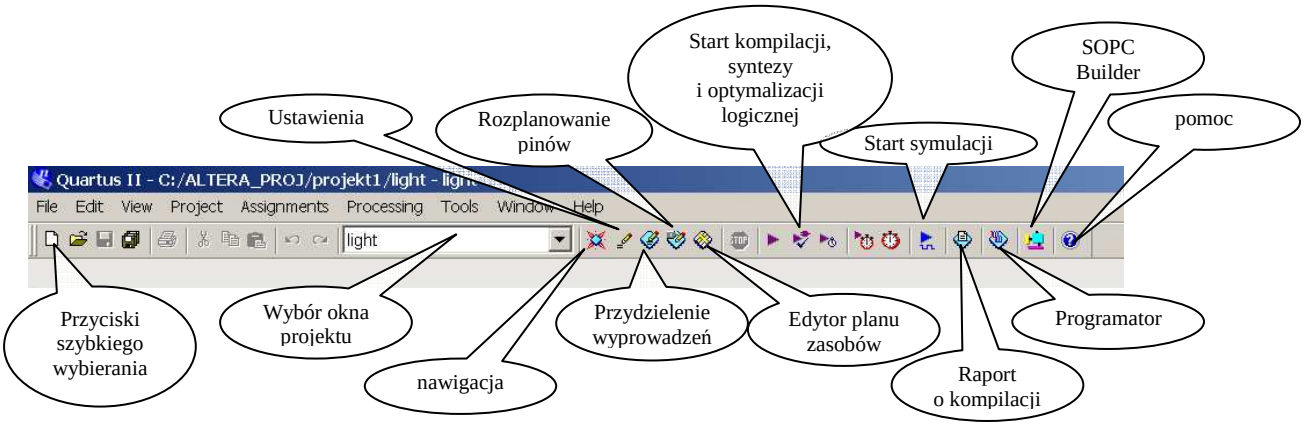


Rys. 2b. *Etapy realizacji układu w strukturze programowalnej PLD*

II. Wprowadzenie do systemu projektowego Quartus

1. Wstęp

W instrukcji przedstawiono krok po kroku tworzenie nowego projektu w środowisku Quartus II ver.8.1 wraz z krótkimi komentarzami dotyczącymi podstawowych ustawień konfiguracyjnych. Wymieniono także najczęściej wykorzystywane podczas zajęć laboratoryjnych elementy biblioteczne omawianego środowiska. Quartus dostarcza obszerną pomoc. Dostęp do pomocy można uzyskać korzystając z menu *Help* > *Contents* lub po wciśnięciu przycisku **F1**. Pakiet projektowy Quartus II obsługuje zestaw dydaktyczny ALTERA_DE 1. Na *rys. 3.* przedstawiono menu główne środowiska Quartus II wraz z opisem podstawowych elementów.



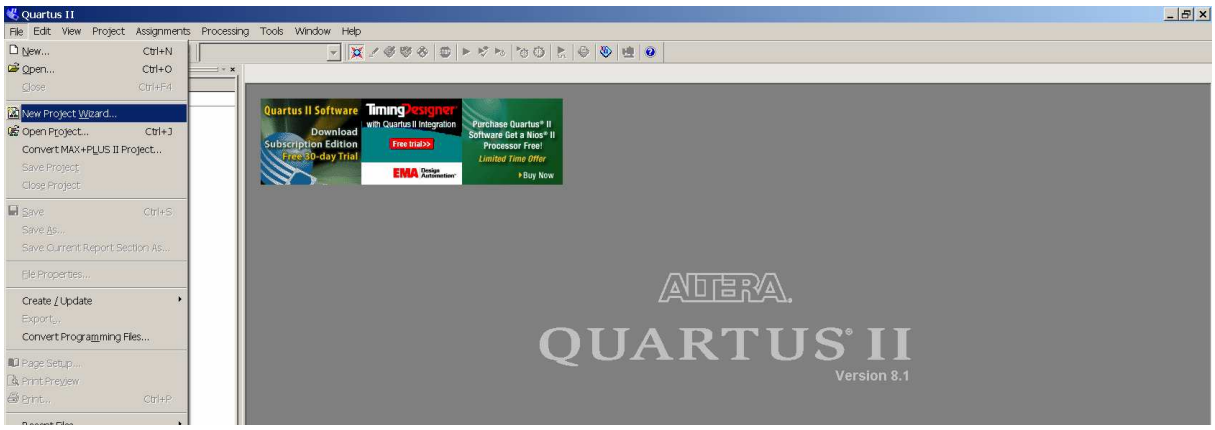
Rys. 3. *Menu główne programu* Quartus II

W środowisku Quartus II możliwe jest wykonanie następujących czynności:

- wprowadzenie i edycja projektu,
- kompilacja projektu,
- określenie docelowego układu programowalnego,
- przyporządkowanie wyprowadzeń,
- symulacja czasowa i funkcjonalna,
- programowanie układu.

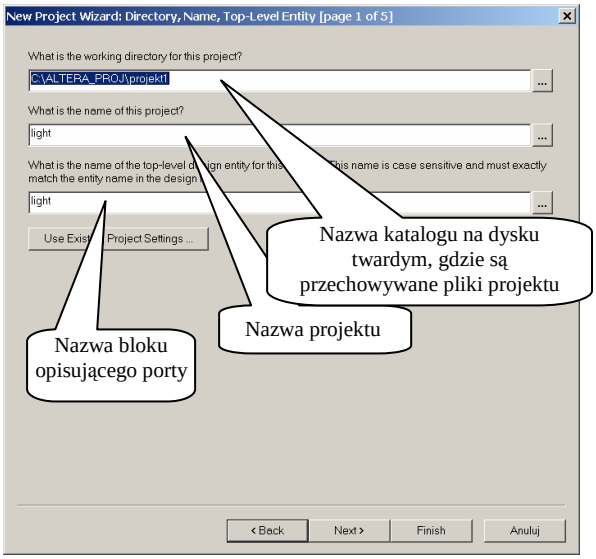
2. Tworzenie nowego projektu

Projekty w programie Quartus II mają budowę hierarchiczną. Jeden blok programu może zawierać inne. Tworzenie nowego projektu zostanie zaprezentowane z wykorzystaniem kreatora *New Project Wizard (rys.4a.)* **Pracę nad nowym projektem rozpoczyna się od utworzenia na dysku twardym katalogu, w którym zostaną umieszczone pliki składowe projektu.**



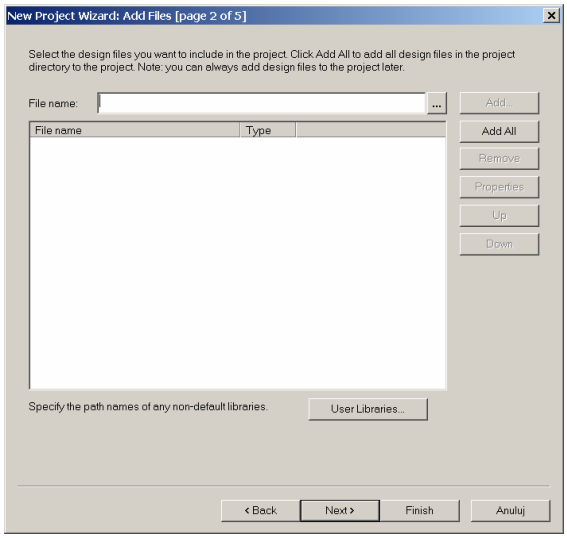
Rys. 4a. *Uruchomienie kreatora New Project Wizard*

1. Wybieramy z menu *File > New Project Wizard*. Uzyskane okno dialogowe w przyszłości możemy pominąć zaznaczając opcje: *Don't show me this introduction again.* Po wybraniu opcji *Next* uzyskamy okno umożliwiające **definiowanie katalogu i nazwy dla nowego projektu** (*rys. 4b*). Należy wpisać katalog roboczy zgodnie z zaleceniami prowadzącego zajęcia laboratoryjne.



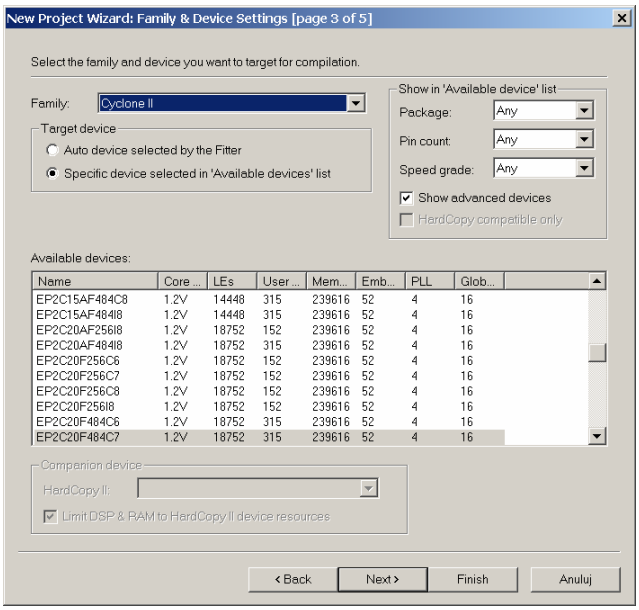
Rys. 4b. *Wprowadzenie katalogu i nazwy projektu*

2. Kolejne okno dialogowe (*rys. 5*) umożliwia dodanie do tworzonego projektu istniejących plików. W naszym przypadku nie mamy do dyspozycji gotowego pliku źródłowego – wybieramy *Next*.



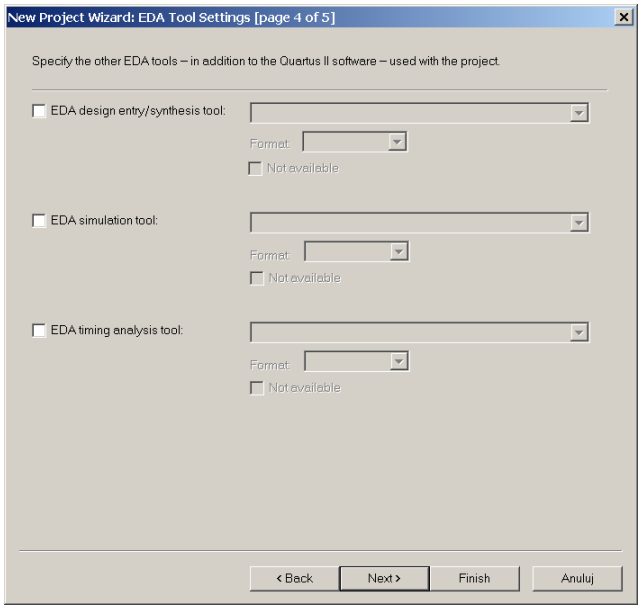
Rys. 5 *Dodanie do projektu istniejących plików*

3. Okno dialogowe (*rys. 6*) umożliwia wybór rodziny układów programo – walnych, a następnie konkretnego układu, na którym dokonana będzie implementacja. W naszym przypadku należy wybrać rodzinę układów programowalnych FPGA Cyclone II i układ EP2C20F484C7. Konfiguracja ta równoważna jest z wybraniem struktury FPGA, która wchodzi w skład zestawu edukacyjnego Altera_DE 1.

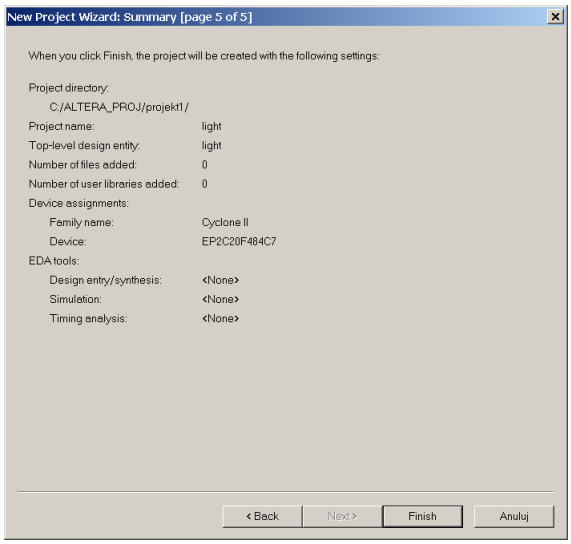


Rys. 6. *Wybór układu programowalnego*

4. W kolejnym oknie dialogowym (*rys. 7*), chcąc wyłącznie korzystać z narzędzi oferowanych w ramach środowiska Quartus, nie zaznaczając żadnego pola wybieramy *Next*.



Rys. 7 *Specyfikacja narzędzi firm innych niż Altera*



Rys. 8 *Podsumowanie ustawień tworzonego projektu*

5. W ostatnim oknie dialogowym kreatora (*rys. 8*), w którym przedstawiono wybrane ustawienia projektu należy wybrać opcję *Finish*.

3. *Opis układu cyfrowego*

System Quartus II umożliwia przygotowanie opisu układu cyfrowego w postaci:

- pliku tekstowego w języku opisu sprzętu: VHDL, VERILOG, AHDL
- **schematu z wykorzystaniem standardowych symboli układów cyfrowych**
- **przebiegów czasowych**.

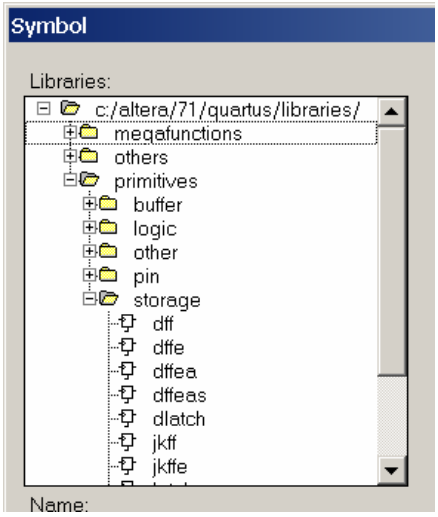
W laboratorium najczęściej będzie wykorzystywany drugi ze sposobów. Podstawowym warunkiem pracy w tym trybie, jest biegłe posługiwanie sie symbolami elementów cyfrowych.

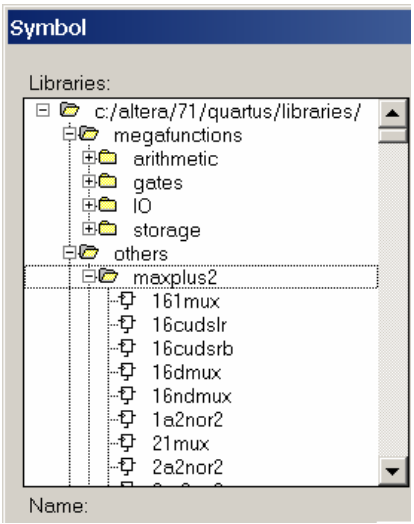
4. *Wybrane elementy biblioteczne dostępne w środowisku* Quartus

Poniżej przedstawiono wybrane elementy biblioteczne dostępne w środowisku Quartus:

- **Podstawowe elementy (*primitives*):**
  - Przerzutniki i zatrzaski (*primitives / storage*):

Opis	Oznaczenie
Przerzutnik typu D	*dff* *dffe*
Przerzutnik typu RS	*srff* *srffe*
Przerzutnik typu JK	*jkffe* *jkff*
Przerzutnik typu T	*tff* *tffe*
Zatrzask	*dlatch*





  - Bramki (*primitives / logic*):

Opis	Oznaczenie
AND	*and2..and12*
NAND	*nand2..nand12*
NOT	*not*
OR	*or2..or12*
NOR	*nor2..nor12*
EX-OR	*xor*
EX-NOR	*xnor*
OR z zanegowanymi wejściami	*bor2..bor12*
NOR z zanegowanymi wejściami	*bnor2..bnor12*
AND z zanegowanymi wejściami	*band2...band12*

  - Piny we / wy (*primitives / pin*):

Opis	Oznaczenie
dwukierunkowy	*bidir*
wejściowy	*input*
wyjściowy	*output*

- Bufory (*primitives / buffers*):

Opis	Oznaczenie
zmiana nazwy linii sygnałowej	*wire*
trójstanowy	*tri*

- inne (*primitives / other*):

Opis	Oznaczenie
stała	*constant*
masa sygnałowa (zero logiczne)	*gnd*
zasilanie (jedynka logiczna)	*VCC*

- **Makromoduły (*others / maxplus2)***

  - układy z rodziny 74xxx
  - sumatory
  - bufory
  - rejestry
  - rejestry przesuwające
  - zatrzaski
  - jednostki ALU
  - komparatory
  - konwertery
  - liczniki
  - dekodery
  - kodery
  - dzielniki częstotliwości
  - multipleksery

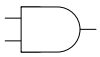
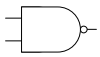
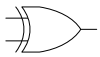
- **Megafunkcje parametryzowane *lpm (megafunction)***

Oznaczenie	Opis
lpm_and *lpm_or* *lpm_xor*	Bramki wielobitowe (*megafunctions/gates*)
lpm_bustri	Wielobitowy bufor trójstanowy (*megafunctions/gates*)
lpm_clshift	Uniwersalny rejestr przesuwający z przesuwaniem logicznym, arytmetycznym lub obrotem (*megafunctions/gates*)
lpm_constans	stała (*megafunctions/gates*)
lpm_decode	Dekoder 1 z N (*megafunctions/gates*)
lpm_inv	Wielobitowy inwerter (*megafunctions/gates*)
busmux	Multiplekser grupowy „2 na 1” (*megafunctions/gates*)
lpm_ mux	Multiplekser grupowy „N na M” (*megafunctions/gates*)
mux	Pojedynczy multiplekser „N na 1” (*megafunctions/gates*)
lpm_abs	Funkcja wartości bezwzględnej (*megafunctions/arithmetic*)
lpm_add_subb	Wielobitowy sumator/subtraktor (*megafunctions/arithmetic*)
lpm_compare	Komparator dwóch liczb n-bitowych (*megafunctions/arithmetic*)
lpm_counter	Wielobitowy licznik z różnymi wariantami sterowania (*megafunctions/arithmetic*)
lpm_mult	Wielobitowy multiplikator (*megafunctions/arithmetic*)
divide	Wielobitowy układ dzielący (*megafunctions/arithmetic*)
lpm_ram_dq	RAM z rozdzielonymi portami zapisu i odczytu (pamięć dwuportowa) (*megafunctions/storage*)
lpm_ram_io	RAM z jednym portem dwukierunkowym (*megafunctions/storage*)
lpm_rom	ROM (tablica stałych wartości) (*megafunctions/storage*)
csdpram	Pamięć RAM jednocyklowa z oddzielonymi portami wejścia-wyjścia (*megafunctions/storage*)
csfifo	Pamięć FIFO (*megafunctions/storage*)

5. Specyfikacja projektu w edytorze graficznym

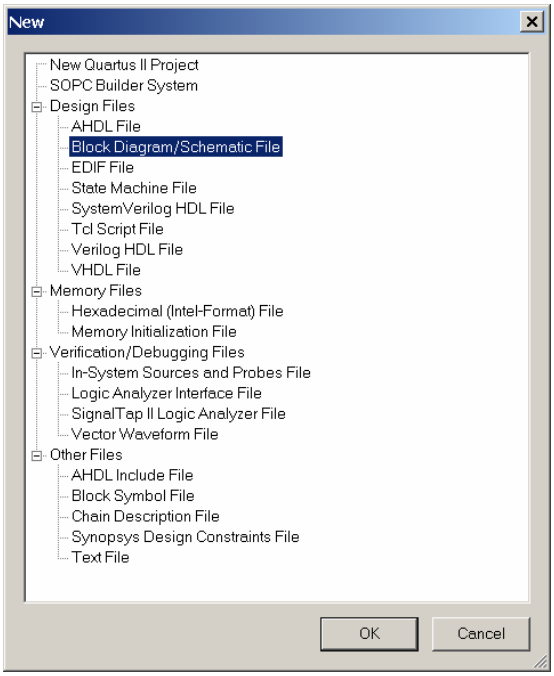
Jako pierwszy z projektów wybrano ćwiczenie mające głównie na celu utrwalenie zasady działania podstawowych bramek logicznych. Poniżej przedstawiono tablice prawdy dla poszczególnych bramek.

Tablice prawdy dla bramek: NOT, AND, OR, NAND, NOR, EXOR, EXNOR

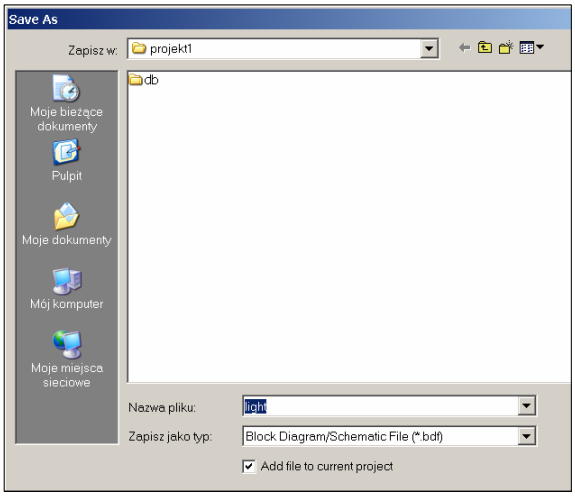
p	q	NOT p	NOT q	p AND q	p OR q	p NAND q	p NOR q	p EXOR q	p EXNOR q
0	0	1	1	0	0	1	1	0	1
0	1	1	0	0	1	1	0	1	0
1	0	0	1	0	1	1	0	1	0
1	1	0	0	1	1	0	0	0	1
									

W celu uruchomienia edytora graficznego wybieramy: *File > New*.

1. W oknie dialogowym (*rys. 9*) należy wybrać *Block Diagram/Schematic File*.
2. Schemat zapisujemy komendą *File > Save As*, pod nazwą „light”, **w katalogu podanym uprzednio przez prowadzącego**. Przed zapisaniem pliku należy wybrać opcję *Add file to current project* (*rys. 10*).

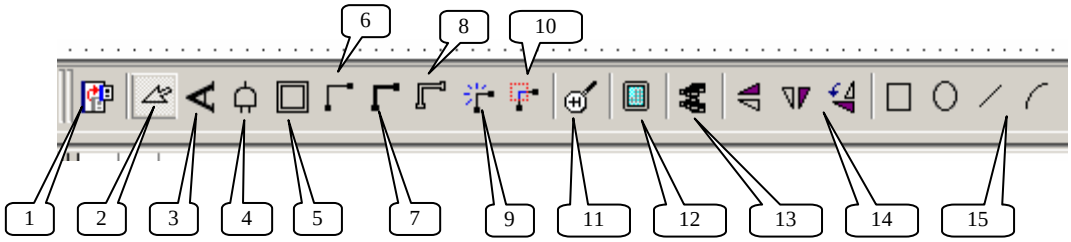


Rys. 9 *Wybór sposobu opisu układu cyfrowego*



Rys. 10 *Zapis tworzonego schematu*

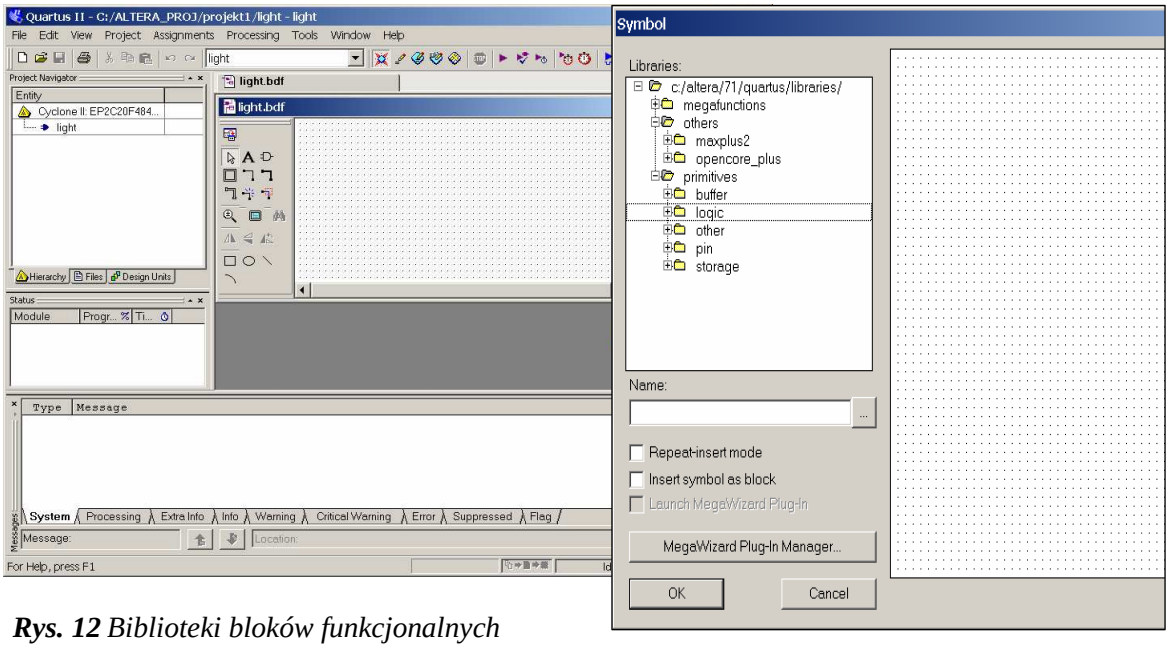
Pracą edytora graficznego steruje się za pomocą narzędzi dostępnych na pasku narzędziowym *(rys. 11)*, który jest domyślnie widoczny po lewej stronie okna edytora. Zawiera on ikony następujących narzędzi:



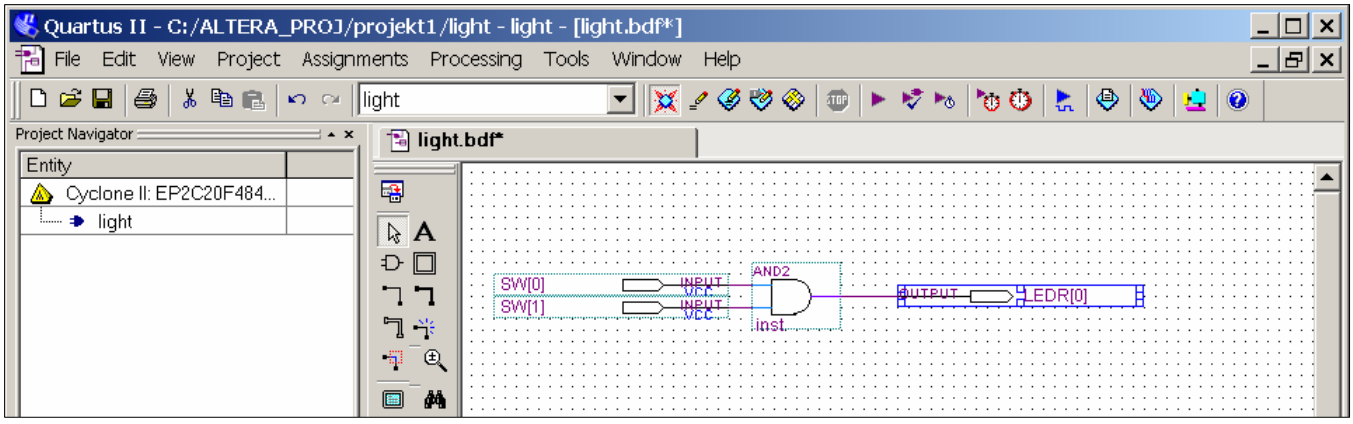
***Rys 11**. Pasek narzędzi w edytorze graficznym*

1. *Detach Window* (Detach the window from Quartus II) – pozwala na odłączenie lub dołączenie okna edytora do ramek nawigatora projektu.
2. *Selection Tool* (Selects objects and text) – narzędzie zaznaczania i chwytania elementów schematu.
3. *Text Tool* (Edits and inserts text) – włączenie narzędzia do wpisywanie i edycji tekstu na schemacie.
4. *Symbol Tool* (Inserts one or more copies of a symbol) – otwiera okno wprowadzania symboli z biblioteki – okno to może też być otwierane poprzez dwukrotne kliknięcie na pustym obszarze schematu.
5. *Block Tool* (Inserts one or more blocks) – pozwala na wprowadzanie bloków do schematu.
6. *Orthogonal Node Tool* (Draws orthogonal node lines) – narzędzie do rysowania pojedynczych połączeń – załącza się automatycznie w przypadku rozpoczęcia rysowania od „wiszącego” wyprowadzenia.
7. *Orthogonal Bus Tool* (Draws orthogonal bus lines) – narzędzie do rysowania wirtualnej szyny połączeniowej.
8. *Orthogonal Conduit Tool* (Draws orthogonal conduit lines) – narzędzie do rysowania wirtualnego przewodu połączeniowego.
9. *Use Rubberbanding* – włącz automatyczne rysowanie połączeń przy użyciu myszy.
10. *Use Partial Line Selection* – umożliwia zaznaczanie fragmentu połączenia poprzez zaznaczenie obszaru – podwójne kliknięcie na linię zaznacza całe połączenie.
11. *Zoom Tool* – powiększa (lewy klawisz myszy) lub zmniejsza (prawy klawisz myszy) widoczny fragment schematu.
12. *Full Screen* – włącza tryb pełnoekranowy edytora graficznego – zamknięte zostają pozostałe obszary okna nawigatora projektu.
13. *Find* (Searches for specified text) – narzędzie do wyszukiwania elementów o podanej nazwie lub wprowadzonego tekstu do schematu.
14. Ikony lustrzanego odbicia i obracania elementu na schemacie.
15. Ikony służące do rysowania dodatkowych elementów graficznych: prostokąta, elipsy, linii prostej i łuku.

3. W celu wprowadzenia symboli elementów tworzonego schematu klikamy dwukrotnie lewym przyciskiem myszy na pole edytora i wybieramy żądany blok funkcjonalny *(rys. 12).*
4. Pierwszą badaną bramką jest bramka AND (*and2*), można ją znaleźć w bibliotece *primitives/logic*. W polu edytora graficznego należy ponadto umieścić dwa wejścia (*input*) oraz jedno wyście (*output*) z biblioteki *primitives/pin*.



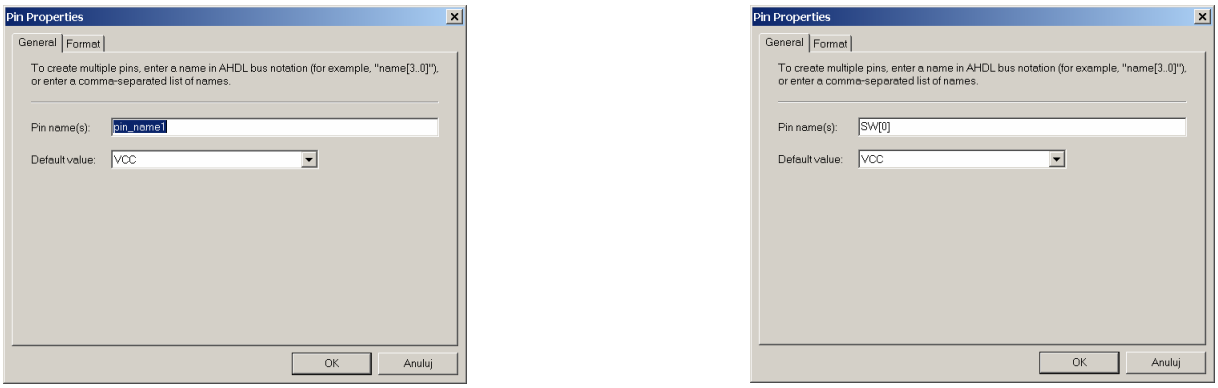
Rys. 12 *Biblioteki bloków funkcjonalnych*



Rys. 13 *Edytor schematów*

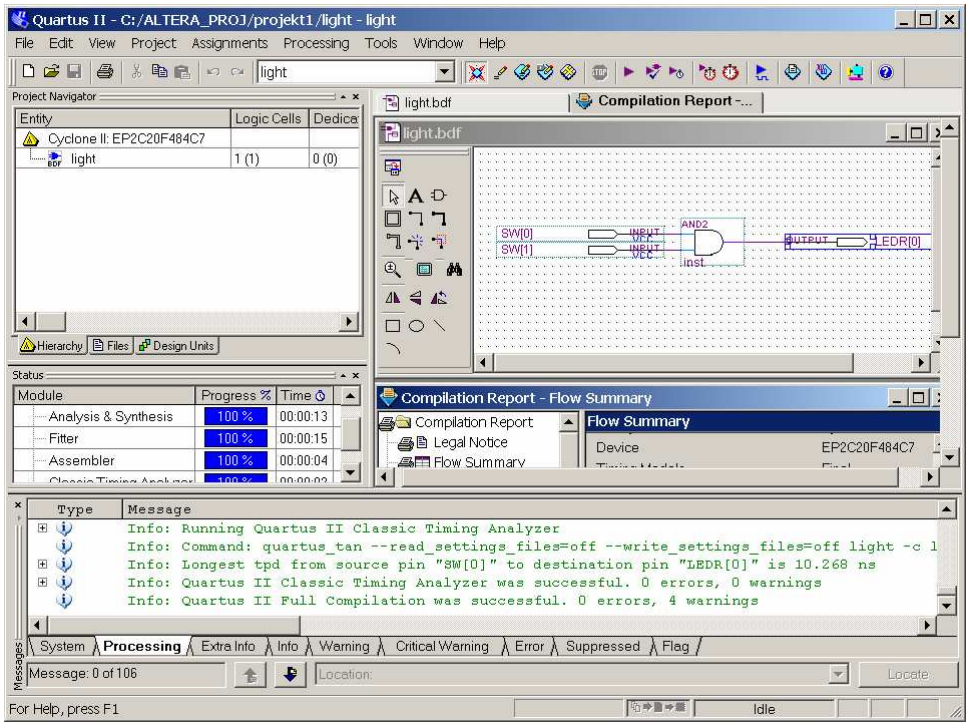
5. Następnie należy wykonać połączenia pomiędzy pinami a badaną bramką *(rys. 13).*

6. Wejściom nadajemy nazwy SW[0] i SW[1], a wyjściu LEDR[0]. Nazwy wprowadzamy poprzez dwukrotne kliknięcie lewym przyciskiem myszy na wejścia i wyjście *(rys. 14).*



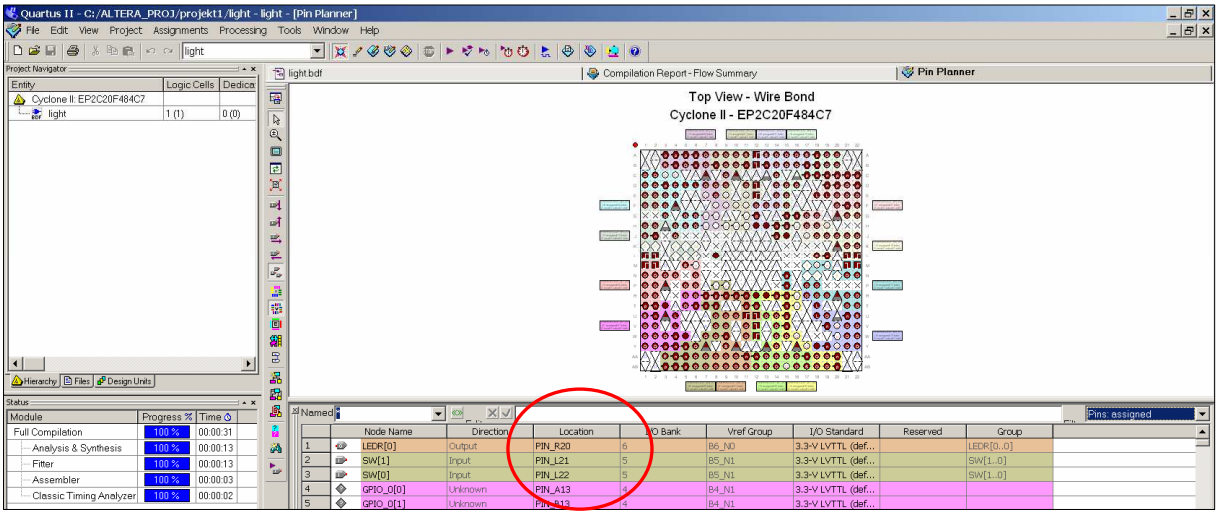
Rys. 14. *Przyporządkowanie nazw pinom we / wy*

7. Kompilujemy stworzony projekt wybierając ikonę . Po kompilacji wyświetla się nam okno dialogowe z podsumowaniem. Po zapoznaniu się z nim można je zamknąć. Po poprawnym zrealizowaniu punktów od 1 do 7 powinniśmy uzyskać zbliżony wygląd okna edytora schematów do przedstawionego na *rys. 15*.



Rys. 15 *Okno edytora schematów po kompilacji*

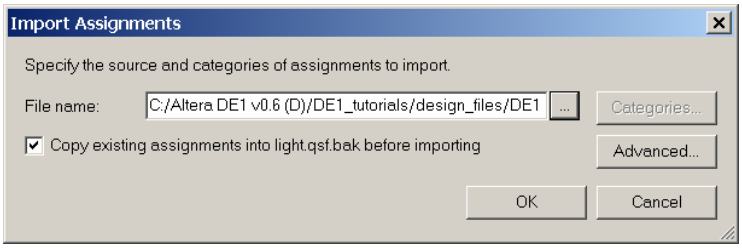
8. Korzystając z informacji zawartych w [1] można przyporządkować wejściom SW[0] – (PIN_L22) i SW[1] – (PIN_L21) oraz wyjściu LEDR[0] – (PIN_R20) fizyczne piny układu *EP2C20F484C7*. Wybieramy *Assignments* > *Pins*, następnie w oknie dialogowym (*rys. 16*), w kolumnie *Location* wybieramy odpowiadający danemu wejściu / wyjściu pin.



Rys. 16. *Przyporządkowanie pinów*

<u>UWAGA!</u> Nieodpowiednie przyporządkowanie pinów układu scalonego może spowodować po uruchomieniu programu uszkodzenie zestawu laboratoryjnego DE1. Kompilator nie zgłosi żadnego ostrzeżenie w tym przypadku. Szczególnie niebezpieczne jest zdefiniowanie pinu będącego wejściem układu scalonego jako pinu wyjściowego.

9. Alternatywnym podejściem do opisywanego powyżej jest zaimportowanie gotowego ich przypisania z pliku: *DE1_pin_assignments.csv*. W pliku tym zdefiniowano wszystkie piny układu ***EP2C20F484C7***, przy czym nadano im nazwy zgodne z nazwami pinów zawartymi w [1]. Kopiujemy w/w plik z katalogu wskazanego przez prowadzącego do katalogu roboczego, a następnie wybieramy z menu *Assignments* > *Import Assignments*. Wskazujemy w/w plik w katalogu roboczym (*rys. 17*) i potwierdzamy klikając *OK*. Po zaimportowaniu przypisania pinów można usunąć poprzednio wprowadzone.



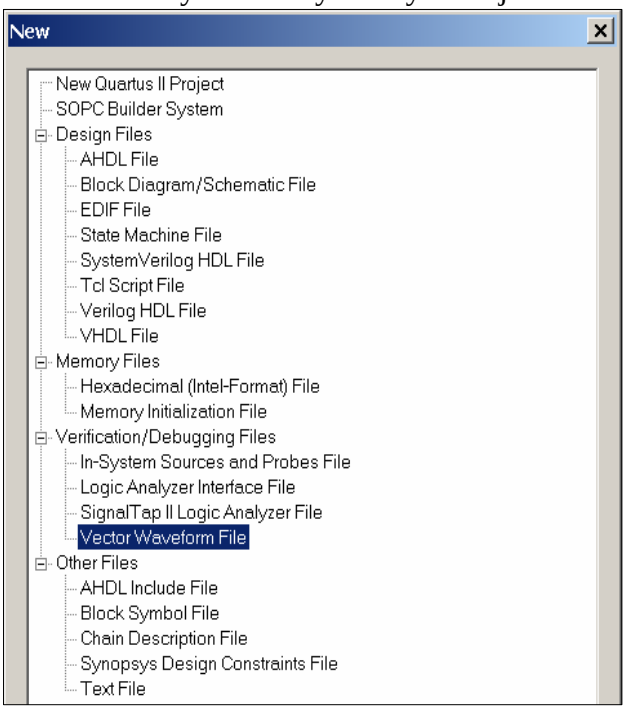
Rys. 17 *Importowanie przyporządkowania pinów*

10. Ponownie kompilujemy stworzony projekt wybierając ikonę: .

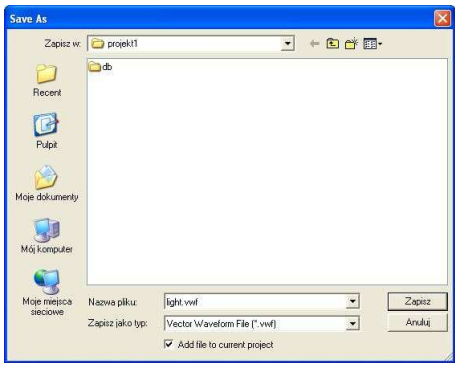
6. Symulacja realizowanego projektu

Symulacja projektowanego układu ma na celu potwierdzenie, że układ został zbudowany prawidłowo. Wykonuje się ją przed zaimplementowaniem projektu w układzie programowalnym. Proces weryfikacji projektu należy rozpocząć od utworzenia pliku zawierającego testowe przebiegi czasowe wszystkich wejść oraz odpowiadające im przebiegi czasowe wszystkich (lub wybranych do analizy) wyjść projektowanego układu, którym należy nadać wartość nie-określoną. Wartości wyjść zostaną określone w trakcie symulacji. Aby utworzyć nowy plik zawierający przebiegi czasowe:

1. Otwieramy okno edytora symulacji – **Waveform Editor** wybierając z menu *File* > *New*, a następnie w oknie dialogowym *(rys. 18) Vector Waveform File***.**



Rys. 18. *Otwieranie nowego pliku Vector Waveform File*



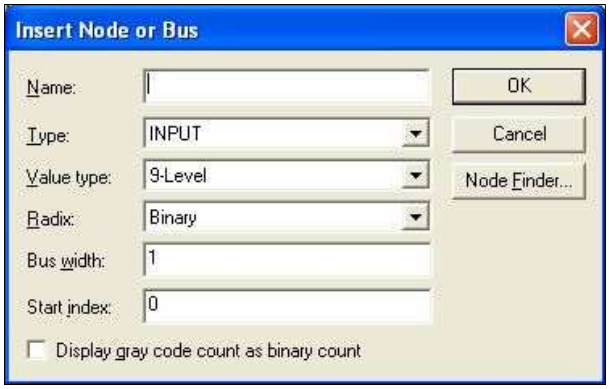
Rys. 19. *Zapisywanie pliku Vector Waveform File*

2. Wybieramy z menu *File* > *Save As* i zapisujemy tworzone wektory testowe (*test vectors*) pod nazwa „Light.vwf” (*rys. 19***).**

3. Przed przystąpieniem do nadawania wartości sygnałom wejściowym należy ustalić czas trwania symulacji oraz rozdzielczość czasową, czyli najkrótszy czas trwania zadanego stanu logicznego. Należy przy tym uwzględnić fakt, że czas całej symulacji można uzależnić od ilości wejść w projektowanym układzie (np. tak aby sprawdzić działanie wszystkich kombinacji wejść) lub od czasu potrzebnego do zamknięcia cyklu pracy układu (np. pełnego cyklu pracy licznika). Przy ustawianiu rozdzielczości czasowej należy pamiętać, że czas propagacji sygnałów może wynosić nawet 30 ns, czyli krótszy sygnał może nie zostać poprawnie zinterpretowany lub skutek jego zmiany może się przesunąć poza następną zmianę stanu. Ustawienie rozdzielczości czasowej następuje po wybraniu polecenia *Edit → Grid Size…*, natomiast czas całej symulacji ustawia się poleceniem *Edit → End Time…* W podanym przykładzie są dwa sygnały wejściowe, zatem rozdzielczość *Grid Size* została ustawiona na 50 ns, a czas całej symulacji *End Time* na 200 ns.

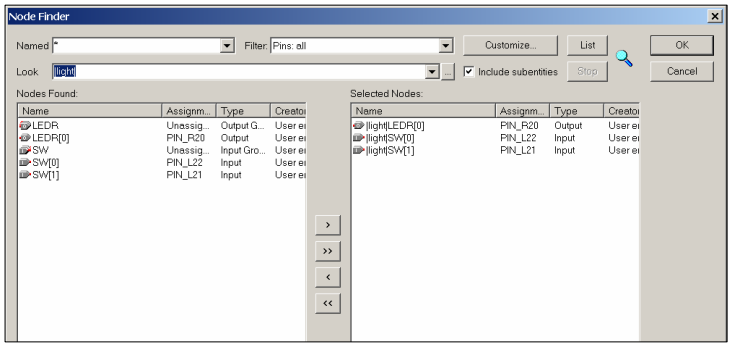
4. Chcąc obserwować przebieg symulacji dla ustawionego okresu symulacji wybieramy z menu *View > Fit in Window*.

5. Wprowadzamy interesujące nas porty we/wy. Klikamy dwukrotnie w kolumnę *Name* (lewa strona edytora) otwierając okno *Insert Node or Bus* (*rys. 20*).



Rys. 20. *Wprowadzanie portów we/wy*

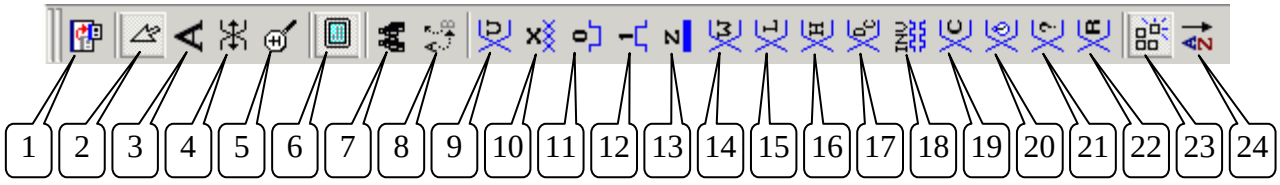
Następnie wybieramy *Node Finder* i w zakładce *Filter* wybieramy *Pins: all*. Klikamy na przycisk *List* i wybieramy z listy port wyjściowy LEDR[0] oraz porty wejściowe SW[0] i SW[1] klikając w przycisk > (*rys. 21*). Pozostałe LEDR i SW są magistralami.



Rys. 21. *Wybór portów we / wy*

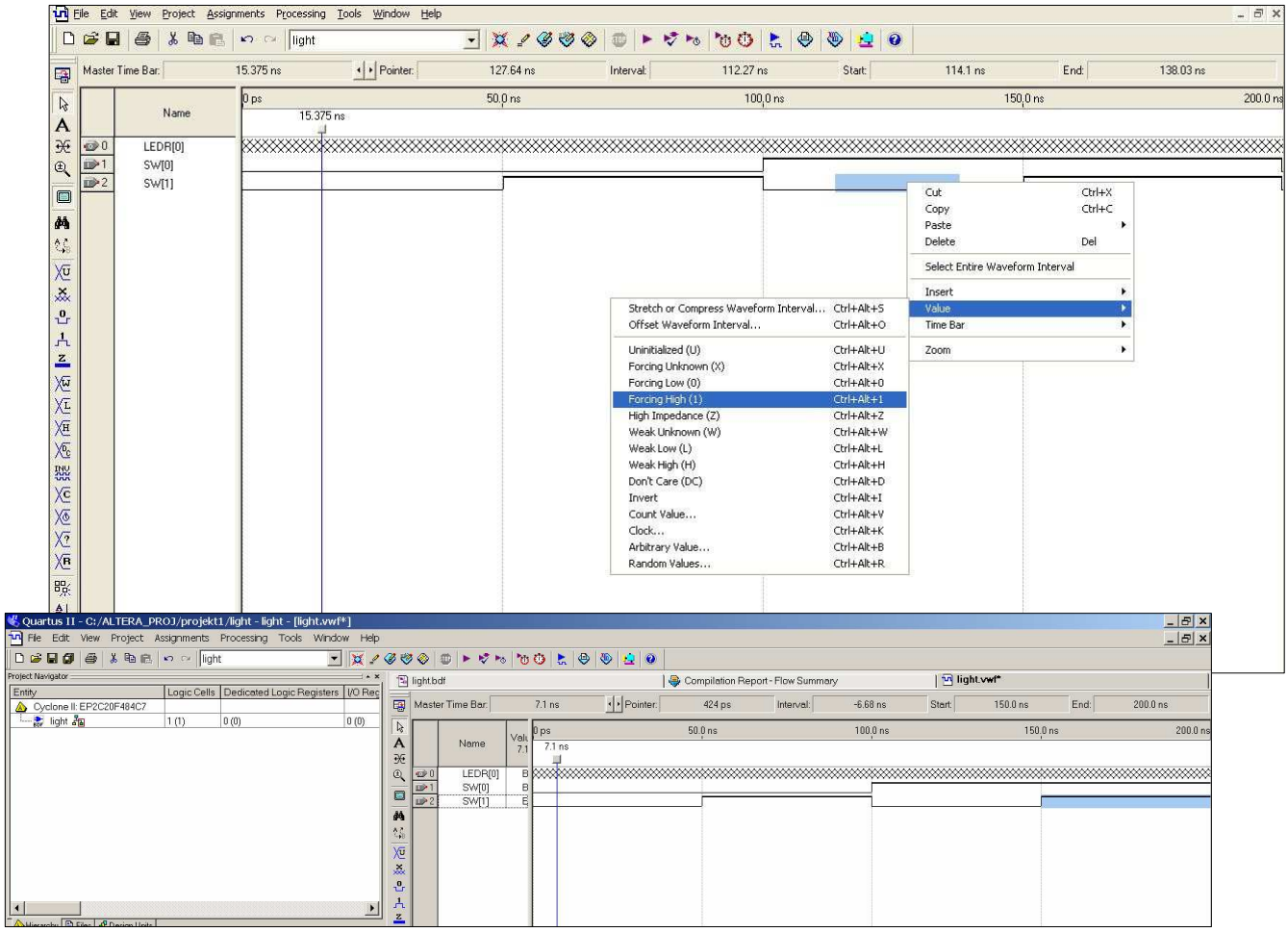
6. Stany portów wejściowych edytujemy zaznaczając myszką wymagany przedział czasu, a następnie klikamy prawym przyciskiem myszy i z menu *Value* wybieramy żądany stan / typ sygnału. Chcąc np. ustawić wartość jedynki logicznej należy wybrać *Value >Forcing High (1)* (*rys. 23*) lub wybrać ikonę  . Należy np. ustawić wartość logiczną „1” dla przedziału czasu od 100 ns do 200 ns dla portu SW[0] oraz dla przedziałów od 50 ns do 100 ns i od 150 ns do 200 ns dla portu SW[1].

Na *rys.22* przedstawiono narzędzia dostępne w edytorze przebiegów czasowych. Umożliwiają one ustawienie poziomu logicznego na wejściu układu.



Rys. 22. *Narzędzia dostępne w edytorze przebiegów czasowych*

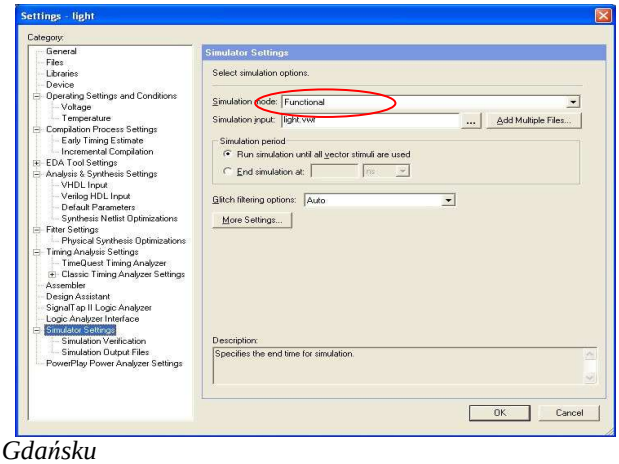
1. *Detach Window* (Detach the window from Quartus II) – pozwala na odłączenie lub dołączenie okna edytora do ramek nawigatora projektu.
2. *Selection Tool* (Selects objects and text) – narzędzie zaznaczania fragmentów lub całości przebiegów czasowych.
3. *Text Tool* (Edits and inserts text) – włączenie narzędzia do wpisywanie i edycji tekstu na przebiegach.
4. *Waveform Editing Tool* – automatycznie zmienia stan sygnału na zaznaczonym obszarze.
5. *Zoom Tool* – powiększa (lewy klawisz myszy) lub zmniejsza (prawy klawisz myszy) widoczny fragment przebiegów.
6. *Full Screen* – włącza tryb pełnoekranowy edytora przebiegów, zamknięte zostają pozostałe obszary okna nawigatora projektu.
7. *Find* (Searches for specified text) – narzędzie do wyszukiwania elementów o podanej nazwie lub wprowadzonego tekstu do przebiegów.
8. *Replace* – zamienia wskazany tekst nowym tekstem.
9. *Uninitialized* – ustawia wybrany fragment przebiegu lub sygnał jako niezainicjowany.
10. *Forcing unknown* – ustawia wybrany fragment przebiegu lub sygnał jako nieokreślony.
11. *Forcing Low* – ustawia wybrany fragment przebiegu lub sygnał na niski poziom logiczny.
12. *Forcing High* – ustawia wybrany fragment przebiegu lub sygnał na wysoki poziom logiczny.
13. *High Impedance* – ustawia wybrany fragment przebiegu lub sygnał na stan wysokiej impedancji dla wyprowadzeń trójstanowych.
14. *Weak unknown* – ustawia wybrany fragment przebiegu lub sygnał jako nieokreślony, jeśli nie została wymuszona inna wartość.
15. *Weak Low* – ustawia wybrany fragment przebiegu lub sygnał jako niski poziom logiczny, jeśli nie została wymuszona inna wartość.
16. *Weak High* – ustawia wybrany fragment przebiegu lub sygnał jako wysoki poziom logiczny, jeśli nie została wymuszona inna wartość.
17. *Don't Care* – ustawia wybrany fragment przebiegu lub sygnał jako nieistotny dla symulatora.
18. *Inverst* – zmienia stan logiczny wybranego fragmentu przebiegu lub sygnału na przeciwny.
19. *Count Value* – zastępuje stan logiczny wybranego fragmentu przebiegu lub sygnału wartością licznika o zadanych parametrach.
20. *Overwrite Colck* – zastępuje stan logiczny wybranego fragmentu przebiegu lub sygnału przebiegiem prostokątnym o zadanych parametrach.
21. *Arbitrary Value* – zastępuje stan logiczny wybranego fragmentu przebiegu lub sygnału wartością podaną przez użytkownika.
22. *Random Value* – zastępuje stan logiczny wybranego fragmentu przebiegu lub sygnału wartością losową.
23. *Snap to Grid* – zezwala na występowanie zmian stanu logicznego sygnałów zgodnie z podaną rozdzielczością czasową (Grid).
24. *Sort* – sortuje sygnały według nazw i typów.



Rys. 23. *Ustawianie stanów portów wejściowych*

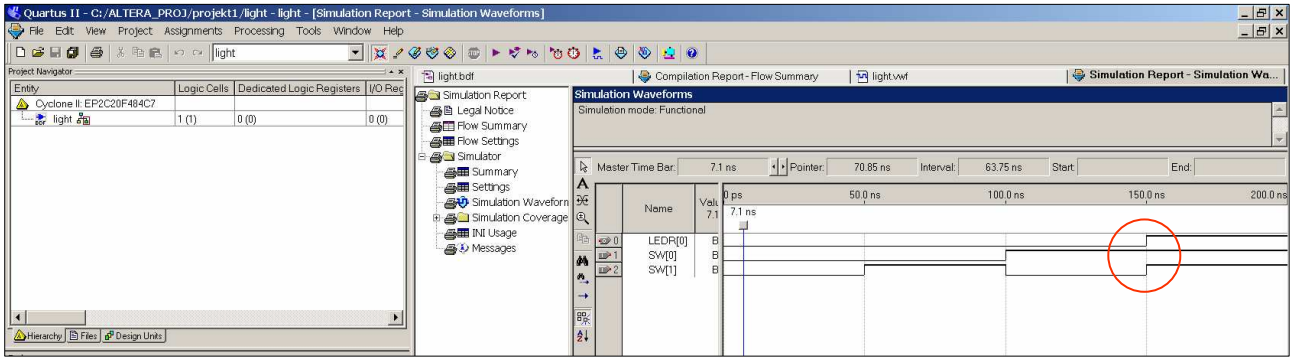
7. Aby wykonać symulacje trzeba wygenerować *Functional Simulation Netlist* wybierając z menu *Processing* > *Generate Functional Simulation Netlist.* Następnie wykonujemy symulacje wybierając z menu *Processing* >*Start Simulation* lub klikając na ikonę  . W środowisku Quartus możemy wykonać dwa typy symulacji:

7.1. **funkcjonalną** – ***functional simulation***, która nie uwzględnia opóźnień propagacyjnych. Typ symulacji ustawiamy wybierając z menu *Assignments* > *Settings*, a następnie dla *Simulator Setings* w polu *Simulation mode* w omawianym przypadku ustawiamy *Functional* (*rys. 24*).



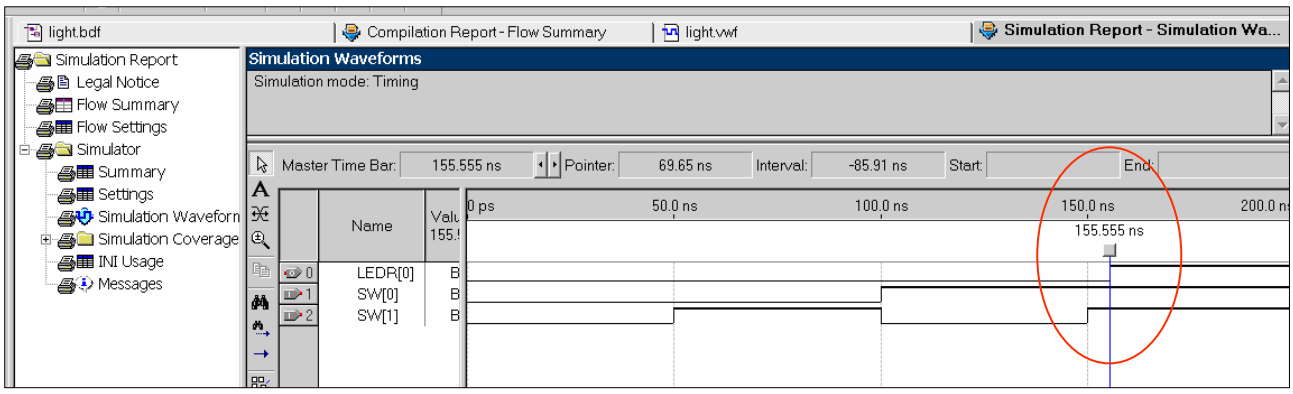
Rys. 24. *Ustawianie typu symulacji*

Po symulacji możemy przystąpić do analizy przebiegu sygnału wyjściowego LEDR[0] *(rys. 25)*. Zmiana stanu na wyjściu następuje bez żadnego opóźnienia w stosunku do chwili, w której wejście SW[1] zmienia po raz drugi stan z „0” na „1” (*rys. 25*).



Rys. 25. *Wynik symulacji funkcjonalnej* ***(functional simulation)***

7.2. czasową – *timing simulation*, która uwzględnia opóźnienia propagacyjne występujące w układzie. W tym przypadku w polu *Simulation mode* ustawiamy ***timing*** (*rys. 24*). Po zrealizowaniu symulacji możemy zauważyć (*rys. 26*) opóźnienie propagacyjne, które występuje w rzeczywistym układzie. Zmiana stanu na wyjściu LEDR[0] następuje po ok. 5,5 ns w stosunku do chwili, w której wejście SW[1] zmienia po raz drugi stan z „0” na „1”.



Rys. 26. *Wynik symulacji czasowej* ***(timing simulation)***

7. Konfiguracja układu FPGA i pamięci flash

Aby umożliwić konfigurację układu FPGA lub pamięci flash znajdujących się na płycie DE_1 należy:

- zestaw DE_1 podłączyć do komputera poprzez złącze USB,
- zainstalować sterownik *USB-Blaster*. Podczas pierwszego podłączenia zasilanego zestawu do komputera system Windows zapyta o lokalizację sterownika, należy wówczas wybrać katalog *quartus/ drivers/ usb-blaster*. Po zainstalowaniu sterownika można przystąpić do programowania układu FPGA.

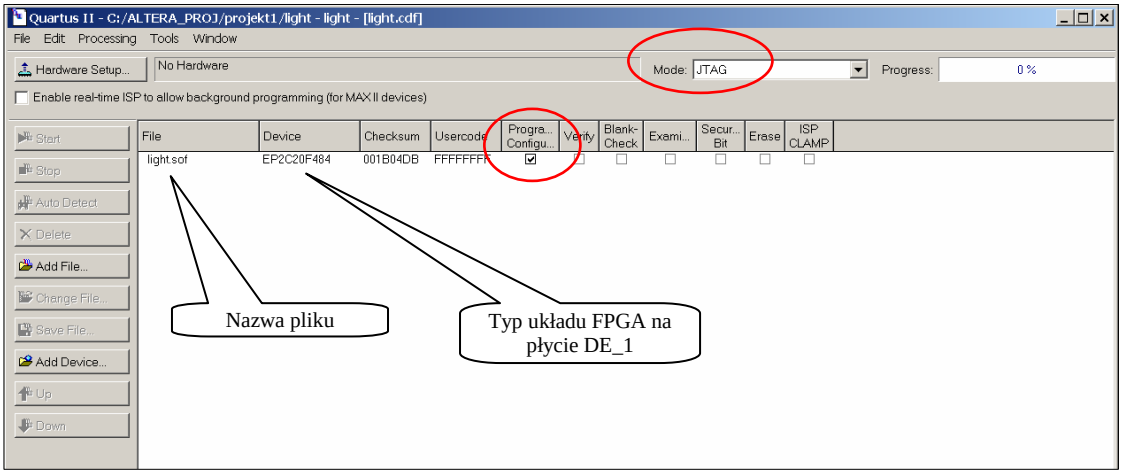
Konfiguracja układu FPGA znajdującego się na płycie DE_1 może być realizowana w dwu trybach:

- *JTAG (Joint Test Action Group)* – w tym przypadku kod wynikowy kompilacji przesyłany jest bezpośrednio do układu FPGA – Cyclone II _ EP2C20F484C7. Układ FPGA zaprogramowany w tym trybie pamięta dane konfiguracyjne tak długo, jak długo włączone jest zasilanie. Zaletą tego rozwiązania jest możliwość przeprowadzenia nieskończonej liczby poprawnych przeprogramowań pamięci SRAM, natomiast wadą utrata programu po wyłączeniu zasilania.
- *AS (Active Serial)* – w tym przypadku układ FPGA wykorzystuje pamięć typu FLASH, do której wpisywane są informacje o konfiguracji. Dane te są pamiętane również po wyłączeniu zasilania. Nie ma potrzeby ponownego konfigurowania układu.

7.1. Programowanie w trybie JTAG

W celu zaprogramowania układu w trybie *JTAG* należy ustawić przełącznik ***RUN/PROG*** znajdujący się na płycie Altera DE_1 w pozycję ***RUN*** (*rys. 27*). Moduł umożliwiający konfigurację wywołujemy wybierając z menu *Tools* > *Programmer* lub  klikając ikonę . W module tym (*rys. 28*) wybieramy tryb programowania *JTAG* oraz sprzęt docelowy *USB-Blaster* (za pomocą polecenia *Hardware Setup).*

Rys. 27*. Wybór trybu programowania za pomocą przełącznika na płycie DE_1*



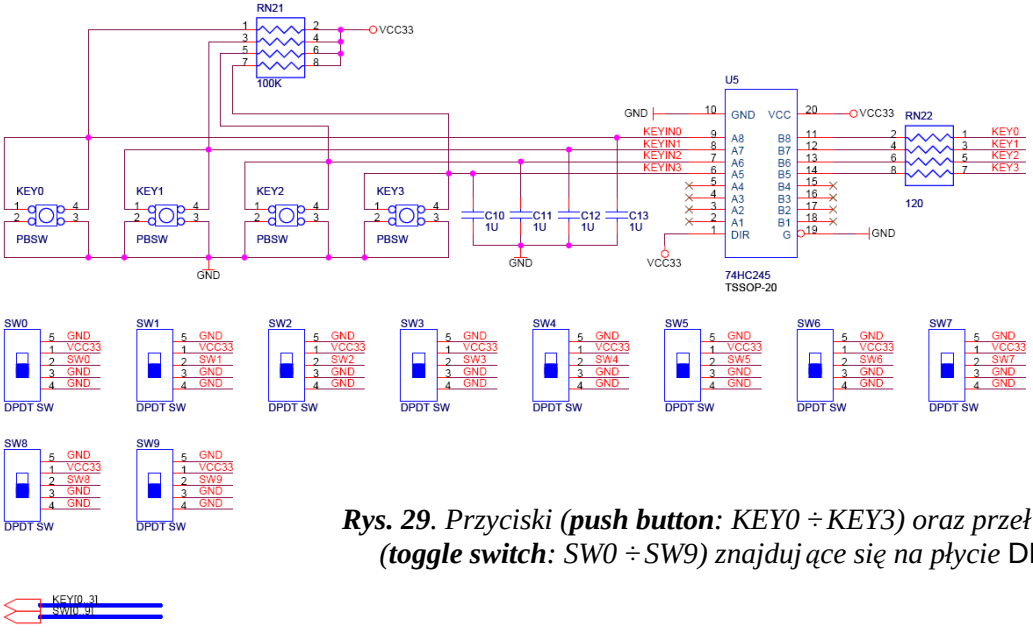
Rys. 28**. Okno dialogowe modułu **Programmer

W polu *File* możemy zauważyć plik o nazwie ***„light.sof”,*** a w polu *Device* znajdujący się na płycie DE_1 układ FPGA – ***EP2C20F484C7*** (*rys. 28*). W przypadku braku na liście wymienionego pliku należy go dodać wybierając *Add File*. W polu dialogowym przedstawionym na (*rys. 28)* należy zaznaczyć opcję w polu *Program/Configure*. Konfigurację układu ***EP2C20F484C7*** rozpoczynamy klikając na *Start* (*rys 28*). Po przeprowadzeniu konfiguracji możemy przystąpić do fizycznego testowania przygotowanego projektu „***light***”.

8. Zestaw dydaktyczny ALTERA DE_1

Do praktycznego testowania realizowanych projektów laboratoryjnych przeznaczony jest zestaw DE_1 firmy *Terasic Technologies* z układem FPGA serii *CycloneII* ***EP2C20F484C7.*** Dokładną instrukcję obsługi zestawu DE_1 zawiera dokumentacja technicznej [1].

8.1. Przyciski i przełączniki



***Rys. 29**. Przyciski (**push button**: KEY0 ÷ KEY3) oraz przełączniki (**toggle switch**: SW0 ÷ SW9) znajdujące się na płycie* DE_1

Do zadawania wejściowych stanów logicznych służą przełączniki SW[0] ÷ SW[9] oraz przyciski KEY[0] ÷ KEY[3]. Stanowi logicznemu **”0” (low).**odpowiada napięcie 0V, a stanowi logicznemu **”1” (high)** – napięcie 3,3V.

Przyciski KEY[0] ÷ KEY[3] (styki normalnie otwarte) są szeregowo połączone z rezystorami wejściowymi RN21 (definiującymi domyślny stan logiczny i wartość prądu płynącego przez styk przycisku) Wciskając przycisk podajemy na odpowiednie wejście układu 74HC245 stan logiczny **”O”** (0V). Układ 74HC245 jest nieodwracającym, trójstanowym buforem cyfrowym (z układem Schmitta). W buforze tym ustalono przepływ sygnału od A do B (wejście DIR podłączono do napięcia zasilania). Wejście G umożliwia wprowadzenie wyjść buforów (B1 ÷ B8) w stan wysokiej impedancji. Na przedstawionym schemacie wejście G jest podłączone na stale do masy zasilania, w tym przypadku bufor jest „przezroczysty” dla sygnałów wejściowych (A1 ÷ A8).Wyjścia buforów (B4 ÷ B7) są podłączone (poprzez rezystory RN22) do pinów układu FPGA.

Signal Name	FPGA Pin No.	Description
SW[0]	PIN_L22	Toggle Switch[0]
SW[1]	PIN_L21	Toggle Switch[1]
SW[2]	PIN_M22	Toggle Switch[2]
SW[3]	PIN_V12	Toggle Switch[3]
SW[4]	PIN_W12	Toggle Switch[4]
SW[5]	PIN_U12	Toggle Switch[5]
SW[6]	PIN_U11	Toggle Switch[6]
SW[7]	PIN_M2	Toggle Switch[7]
SW[8]	PIN_M1	Toggle Switch[8]
SW[9]	PIN_M1	Toggle Switch[9]

Tabela1*. No. pinów FPGA przyporządkowane przełącznikom (**toggle switch**: SW0 ÷ SW9)*

Signal Name	FPGA Pin No.	Description
KEY[0]	PIN_R22	Pushbutton[0]
KEY[1]	PIN_R21	Pushbutton[1]
KEY[2]	PIN_T22	Pushbutton[2]
KEY[3]	PIN_T21	Pushbutton[3]

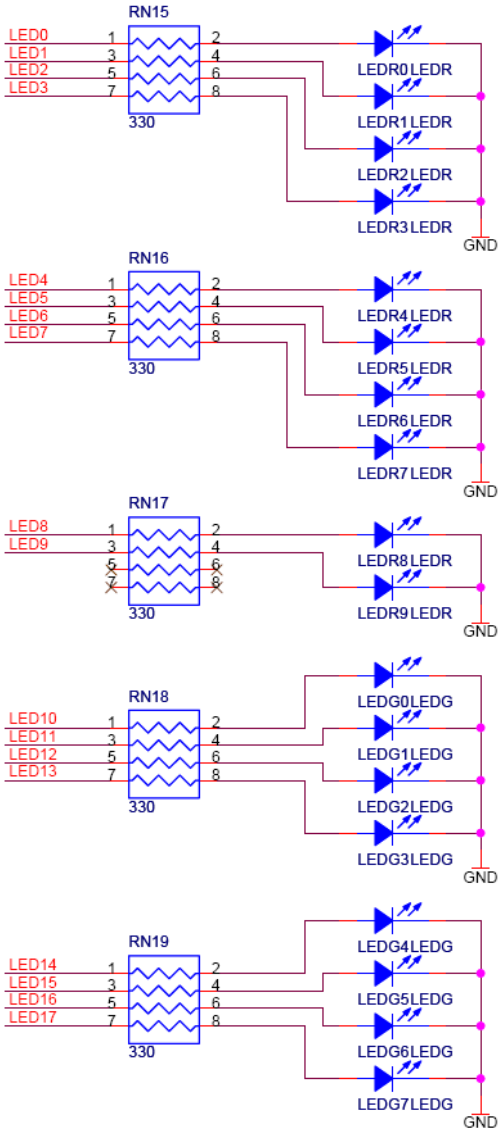
Tabela 2*. No. pinów FPGA przyporządkowane przyciskom (**push button**: KEY0 ÷ KEY3)*

8.2. Diody LED i wyświetlacze 7 segmentowe

Do obserwacji stanów logicznych na wyjściach układu służą diody LED (czerwone i zielone) oraz wyświetlacze typu LED 7– segmentowe. Na *rys. 30* przedstawiono schemat ideowy diod LED znajdujących się na płycie DE_1. Diody LEDR[0] do LEDR[9] (czerwone) i LEDG[0] do LEDG[7]) (zielone) świecą w przypadku wystąpienia na wyjściu układu FPGA stanu logicznego **”1” (3,3V).**

Wyświetlacze 7– segmentowe HEX[0] ÷ HEX[3] są wyświetlaczami ze wspólną anodą (*rys. 31, rys. 32).* Każdy segment wyświetlacza połączony jest poprzez rezystor z pinem układu *Cyclone II* FPGA. Wystąpienie na wyjściu układu FPGA stanu logicznego **”0” (0V**) powoduje zaświecenie segmentu wyświetlacza, natomiast stanu logicznego **”1” (3,3V) –** wyłączenie segmentu.

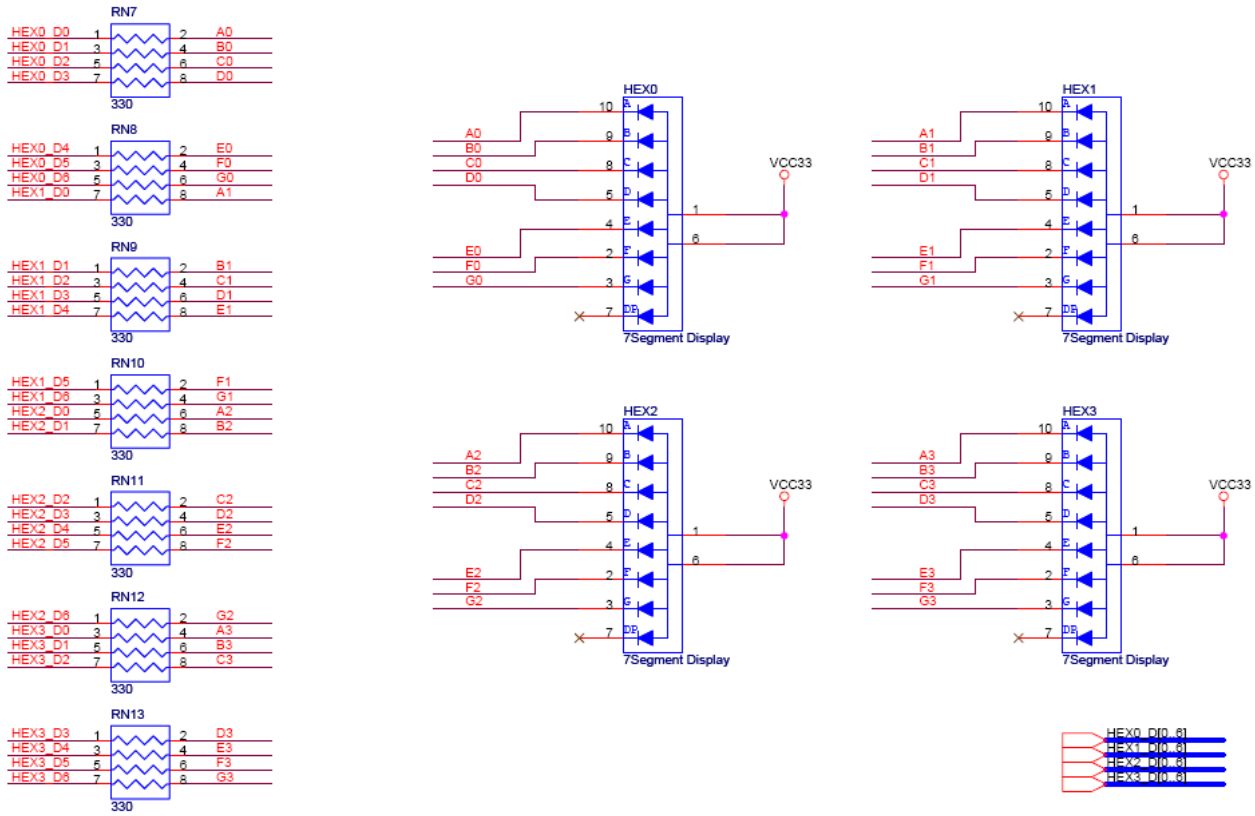
UWAGA! Każdy segment używanego wyświetlacza musi być podłączony!!



***Rys. 30**. Diody LEDR[0] ÷ LEDR[9] i LEDG[0] ÷ LEDG[7] znajdujące się na płycie* DE_1

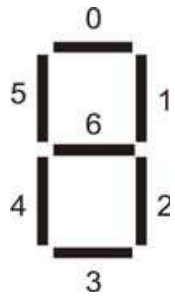
***Tabela 3**. No. pinów FPGA przyporządkowane diodom LED*

Signal Name	FPGA Pin No.	Description
LEDR[0]	PIN_R20	LED Red[0]
LEDR[1]	PIN_R19	LED Red[1]
LEDR[2]	PIN_U19	LED Red[2]
LEDR[3]	PIN_Y19	LED Red[3]
LEDR[4]	PIN_T18	LED Red[4]
LEDR[5]	PIN_V19	LED Red[5]
LEDR[6]	PIN_Y18	LED Red[6]
LEDR[7]	PIN_U18	LED Red[7]
LEDR[8]	PIN_R18	LED Red[8]
LEDR[9]	PIN_R17	LED Red[9]
LEDG[0]	PIN_U22	LED Green[0]
LEDG[1]	PIN_U21	LED Green[1]
LEDG[2]	PIN_V22	LED Green[2]
LEDG[3]	PIN_V21	LED Green[3]
LEDG[4]	PIN_W22	LED Green[4]
LEDG[5]	PIN_W21	LED Green[5]
LEDG[6]	PIN_Y22	LED Green[6]
LEDG[7]	PIN_Y21	LED Green[7]



***Rys. 31**. Wyświetlacze 7 – segmentowe HEX[0] ÷ HEX[3] znajdujące się na płycie* DE_1

Signal Name	FPGA Pin No.	Description
HEX0[0]	PIN_J2	Seven Segment Digit 0[0]
HEX0[1]	PIN_J1	Seven Segment Digit 0[1]
HEX0[2]	PIN_H2	Seven Segment Digit 0[2]
HEX0[3]	PIN_H1	Seven Segment Digit 0[3]
HEX0[4]	PIN_F2	Seven Segment Digit 0[4]
HEX0[5]	PIN_F1	Seven Segment Digit 0[5]
HEX0[6]	PIN_E2	Seven Segment Digit 0[6]
HEX1[0]	PIN_E1	Seven Segment Digit 1[0]
HEX1[1]	PIN_H6	Seven Segment Digit 1[1]
HEX1[2]	PIN_H5	Seven Segment Digit 1[2]
HEX1[3]	PIN_H4	Seven Segment Digit 1[3]
HEX1[4]	PIN_G3	Seven Segment Digit 1[4]
HEX1[5]	PIN_D2	Seven Segment Digit 1[5]
HEX1[6]	PIN_D1	Seven Segment Digit 1[6]
HEX2[0]	PIN_G5	Seven Segment Digit 2[0]
HEX2[1]	PIN_G6	Seven Segment Digit 2[1]
HEX2[2]	PIN_C2	Seven Segment Digit 2[2]
HEX2[3]	PIN_C1	Seven Segment Digit 2[3]
HEX2[4]	PIN_E3	Seven Segment Digit 2[4]
HEX2[5]	PIN_E4	Seven Segment Digit 2[5]
HEX2[6]	PIN_D3	Seven Segment Digit 2[6]
HEX3[0]	PIN_F4	Seven Segment Digit 3[0]
HEX3[1]	PIN_D5	Seven Segment Digit 3[1]
HEX3[2]	PIN_D6	Seven Segment Digit 3[2]
HEX3[3]	PIN_J4	Seven Segment Digit 3[3]
HEX3[4]	PIN_L8	Seven Segment Digit 3[4]
HEX3[5]	PIN_F3	Seven Segment Digit 3[5]
HEX3[6]	PIN_D4	Seven Segment Digit 3[6]

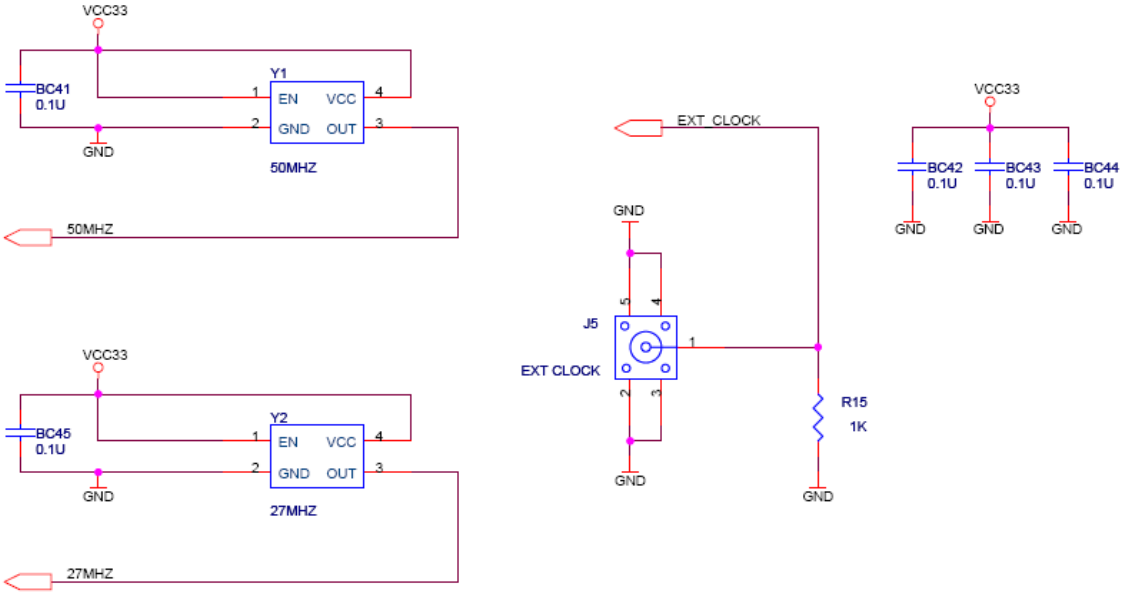


***Rys. 32**. Oznaczenie segmentów wyświetlacza 7 – segmentowego*

***Tabela 4**. No. pinów FPGA przyporządkowane wyświetlaczom 7 – segmentowym*

8.3. Wejścia (zegarowe)

Na płycie DE_1 mamy do dyspozycji trzy oscylatory kwarcowe (*rys. 33)* generujące sygnały o częstotliwości 27 MHz, 24 MHz i 50 MHz. Istnieje również możliwość doprowadzenia sygnału z zewnętrznego generatora



Rys. 33*. Oscylatory kwarcowe znajdujące się na płycie* DE_1

Signal Name	FPGA Pin No.	Description
CLOCK_27	PIN_D12, PIN_E12	27 MHz clock input
CLOCK_50	PIN_L1	50 MHz clock input
CLOCK_24	PIN_A12, PIN_B12	24 MHz clock input from USB Blaster
EXT_CLOCK	PIN_M21	External (SMA) clock input

Tabela 5*. No. pinów oscylatorów kwarcowych*

.

9. Testowanie bramki AND w układzie FPGA

Sprawdź działanie zaimplementowanej bramki AND zadając stany logiczne na jej wejścia za pomocą przełączników SW[0] i SW[1] i obserwując stan na jej wyjściu na diodzie LEDR[0].

10. Pytania kontrolne

1. Wymienić sposoby opisu układu cyfrowego w systemie projektowym Quartus.
2. Jaki jest główny podział elementów bibliotecznych w programie Quartus?
3. Przedstawić tablice prawdy dla bramek NOT, AND, OR, NAND, NOR, XOR.
4. Jaki stan logiczny pojawi się na odpowiednim wejściu układu FPGA po wciśnięciu jednego z przycisków KEY[0] ÷ KEY[3] – odpowiedź uzasadnić na podstawie fragmentu schematu przedstawionego na *rys. 29*.
5. Jaki stan wyjścia układu FPGA spowoduje świecenie diody LED? Na podstawie fragmentu schematu przedstawionego na *rys. 30* oszacować prąd diod LEDR/LEDG.
6. Jaki stan wyjścia układu FPGA spowoduje świecenie segmentu wyświetlacza 7 segmentowego LED? Oszacować prąd płynący przez segment wyświetlacza *(rys. 31).*
7. Dlaczego system projektowy Quartus nie sygnalizuje błędnego przyporządkowania pinów we/wy. Wyjaśnić jakie mogą być negatywne skutki tych błędów?
8. Omówić różnice funkcjonalne między bezpośrednią konfiguracją układu FPGA, a programowaniem pamięci flash.

11. Materiały źródłowe

[1] DE1 Development and Education Board User Manual (DE1_UserManual_v1018.pdf)

[2] Artur Cichowski, Wojciech Śleszyński: Wprowadzenie do systemu projektowego Quartus II PG WEiA 2007 (cwiczenie_1_2_QUATRUS_II.pdf)

[3] Quartus II Introduction Using Schematic Design (tut_quartus_intro_schem.pdf)

[4] Krystyna M. Noga, Marcin Radwański: Projektowanie układów programowalnych w środowisku Quartus II z wykorzystaniem edytora tekstowego, AM Gdynia 2008

[5] http://www.altera.com

[6] P. Zbysiński, J. Pasierbiński: Układy programowalne – pierwsze kroki, BTC 2004

[7] J. Majewski, P. Zbysiński: Układy FPGA w przykładach, BTC 2007