

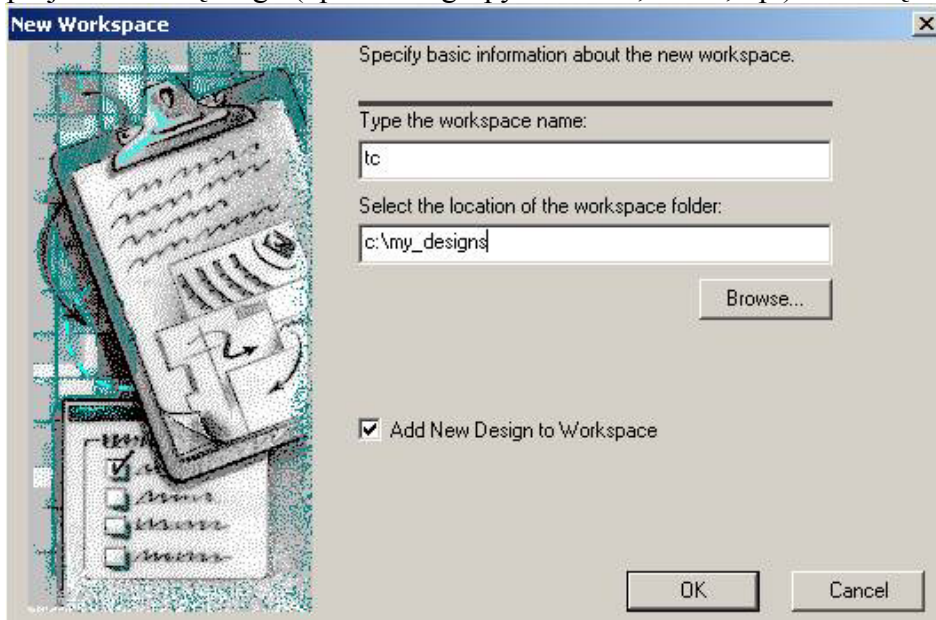
ALGORYTM URUCHOMIENIA I OBSŁUGI PROGRAMU ACTIVE-HDL (zajęcia wprowadzające)

Uruchomienie programu i utworzenie nowego projektu

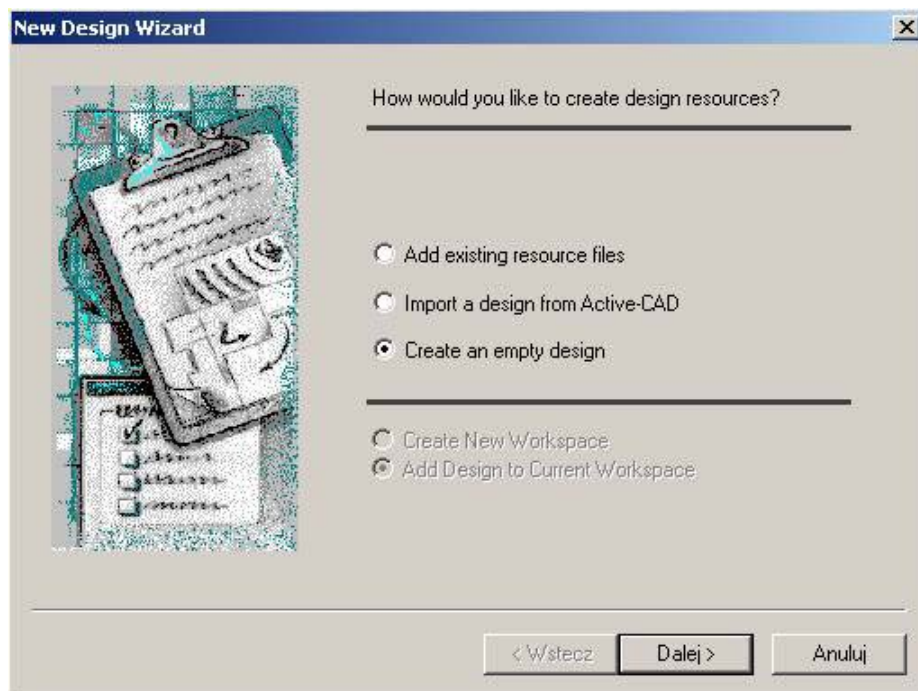
1. Uruchom program Active-HDL 6.1



2. Jeśli po otwarciu nie pojawi się automatycznie okno New Workspace należy go wybrać z paska zadań: **File** i następnie **New, Workspace**. W dalszej kolejności należy określić nazwę projektu nadrzędnego (np. nazwa grupy – wtorek, lub tc, itp.) i wcisnąć **OK**.



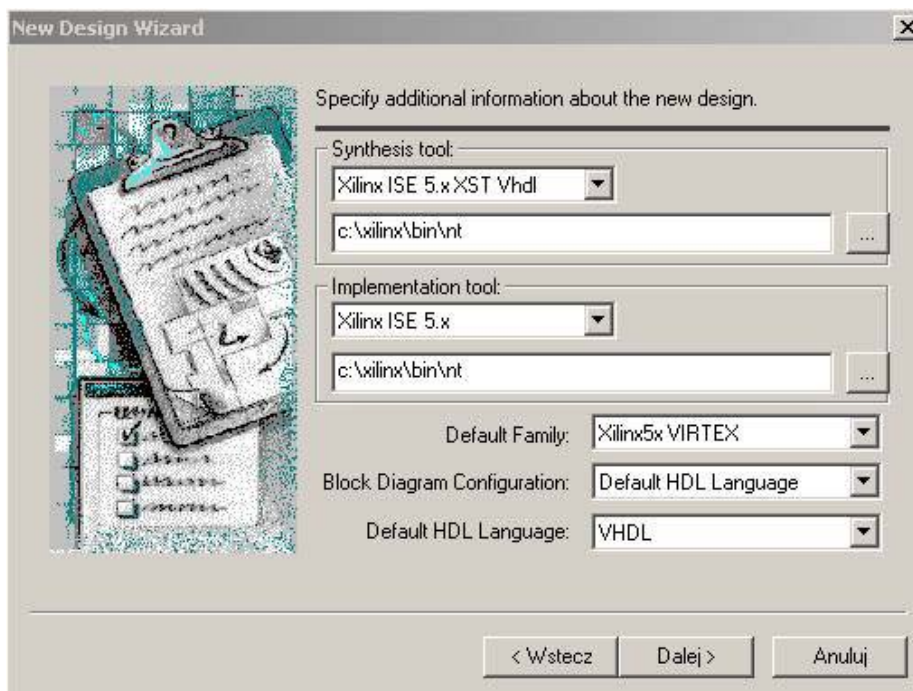
3. Pojawi się okno **New Design Wizard**, gdzie zaznaczamy **Create an empty design**.



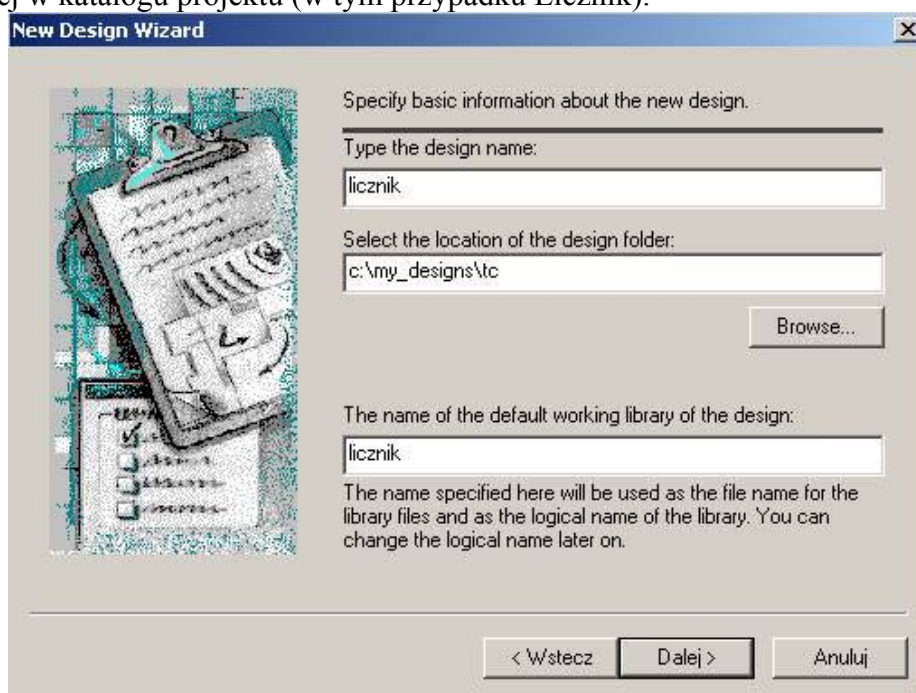
4. Po wybraniu **Dalej** nastąpi określenie narzędzi syntezy i implementacji. W tym miejscu wybór zależy od zainstalowanych opcji (dodatkowego oprogramowania innych firm). Zainstalowanie samego programu Active-HDL uniemożliwia jakiegokolwiek wybór. W pasku zadań: **Synthesis tool** określamy narzędzie syntezy. Pozwala to narysowany schematy wyeksportować do **Veriloga** lub **VHDL** (języków opisu sprzętu). Ponieważ na zajęciach nie będziemy zajmować się syntezą więc można tutaj nic nie wybrać lub wstawić cokolwiek (w zależności od zainstalowanych opcji) np. narzędzie **Xilinx ISE 5.x XST Vhdl**. W pasku zadań: **Implementation Tool** wybieramy gdzie będzie zaimplementowany (do jakiego scalaka) nasz projekt (w zależności od zainstalowanych opcji) natomiast w pasku **Default Family** wybieramy rodzinę układów scalonych do implementacji (w zależności od zainstalowanych opcji).

W pasku **Block Diagram Configuration**: musi być wybrany **EDIF** jeśli nie wybrano narzędzia syntezy.

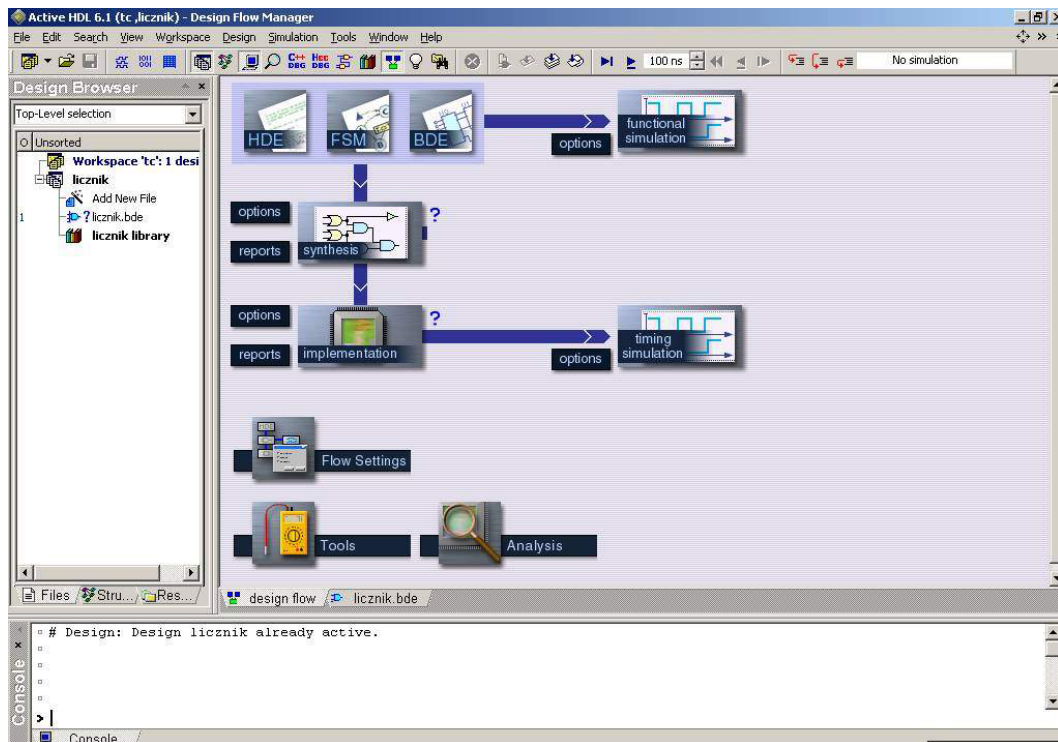
Wybranie EDIF powoduje, że ustawienie w pasku **Default HDL Language** jest nieistotna



5. Po wciśnięciu dalej w okienku **New Design Wizard** określamy nazwę ćwiczenia w wybranym projekcie nadrzędnym. Nazwa ćwiczenia domyślnie określa nazwę biblioteki roboczej w katalogu projektu (w tym przypadku Licznik).



6. Po wybraniu przycisku **Dalej** i następnie **Zakończ** pojawi się okno **Design Flow Manager** (wszystkie jego opcje są dostępne jeśli wybraliśmy wcześniej narzędzie implementacji i syntezy)



Mamy tutaj następujące opcje:

HDE – Edytor tekstowy umożliwiający pisanie w *VHDL* lub *Verilogu*

FSM – edytor graficzny do maszyn stanowych (ale konieczny jest wybór narzędzi syntezy)

BDE – edytor schematów, którego będziemy używać (nie jest konieczny wybór narzędzi syntezy)

Na rysunku brak narzędzia syntezy i implementacji ze względu na brak wyboru.

Flow Settings - umożliwia zmianę wybranych opcji projektu

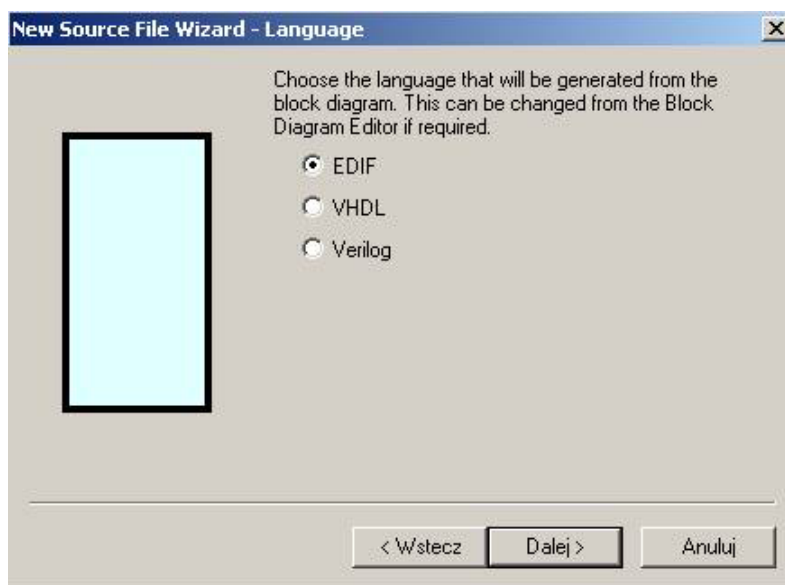
Tools – zaawansowane narzędzia Xilinx do niskopoziomowych operacji (możliwość edycji na poziomie struktury układu FPGA) – nie używamy na zajęciach

Analysis – zaawansowane narzędzia Xilinx do niskopoziomowych operacji (możliwość analizy np. statycznej analizy czasowej) – nie używamy na zajęciach

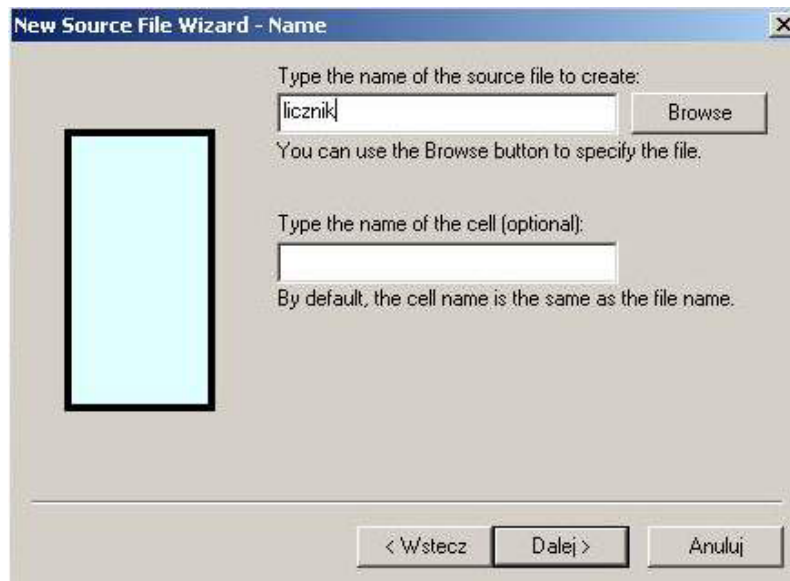
7. Po wybraniu **BDE** zakładamy plik w którym będziemy rysować schemat. W tym celu w następnym oknie wybieramy *Add the generated file to the design*, wciskamy **Dalej**.



Pojawi się wówczas okno *New Source File Wizard – Language*, wybieramy język do którego będzie wyeksportowany nasz schemat (u nas **EDIF** – nie przechodzimy przez syntezę).



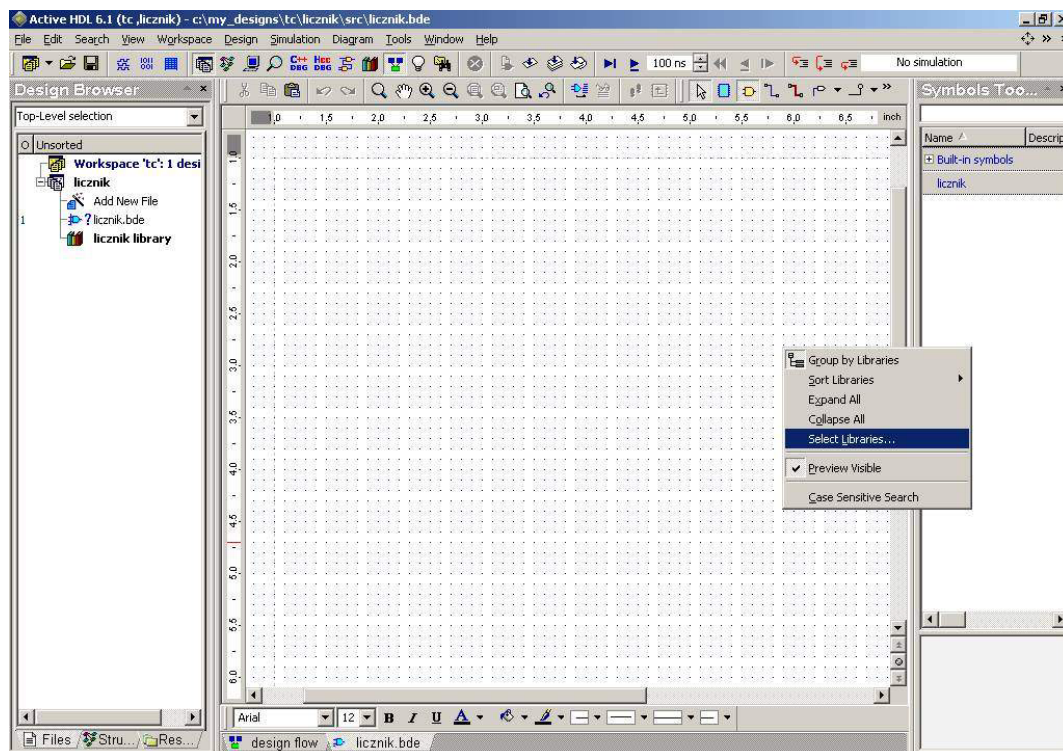
8. Po wciśnięciu **Dalej** w oknie wybieramy w oknie *New Source File Wizard – Name* nazwę pliku w którym będzie zawarty nasz projekt (u nas wpisujemy licznik).



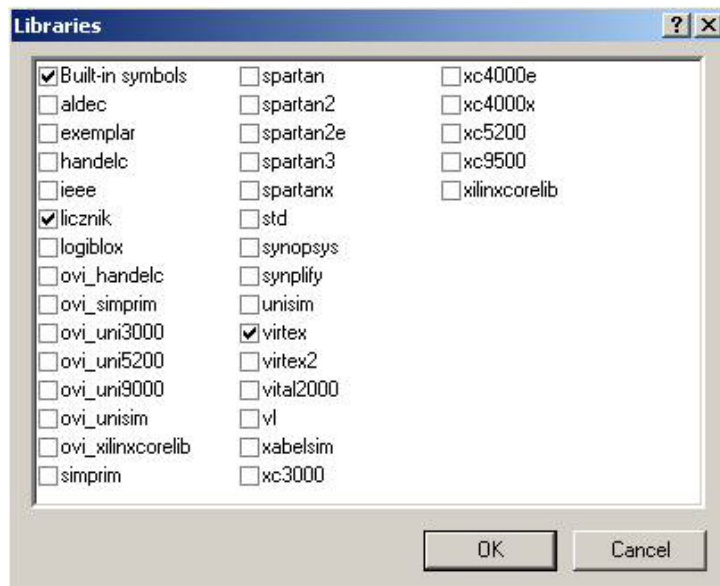
9. Po wciśnięciu **Dalej** mamy możliwość zdefiniowania wejść i wyjść projektowanego układu **New Source File Wizard – Ports** (jeśli na tym etapie wiadomo jakie będą). Zakładamy, że ich nie znamy, więc wybieramy **Zakończ**.

Rysowanie Schematu

1. Po uruchomieniu pojawi się okno do rysowania schematu:



2. Po wybraniu z paska zadań **View/Symbol Toolbox** - (klawisz skrótu **S** albo żółta bramka na drugim od góry pasku zadań) pojawi się okno w którym definiujemy nazwę biblioteki technologicznej z której będziemy wybierać elementy. Po naciśnięciu prawego przycisku myszy na **Built-In Symbols** (w oknie **Symbols Toolbox**) wybieramy **Select Libraries** i **Virtex**.

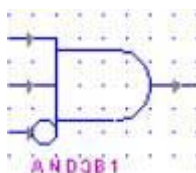


Po wybraniu **Virtex** pojawią się dostępne elementy do symulacji. W górnej linijce poprzez wybranie początkowych liter możemy wyszukiwać elementy w bibliotece.

Po zainstalowaniu Xilinx możliwy jest dostęp do dokumentacji do tych elementów (tzw. Library guide)

3. Rysujemy schemat - podstawy oraz kilka przydatnych wiadomości:

- Po wybraniu elementu z biblioteki wciśniętym lewym klawiszem myszki przeciągamy go na odpowiednie miejsce schematu. Czynność powtarzamy dla pozostałych elementów: **GND**, **VCC**. Do projektu należy używać jedynie elementów wybranej biblioteki!!! Nie należy korzystać z elementów biblioteki **Built-In Symbols**.
- Dla wygody i czytelności projektu użyte elementy można dowolnie ustawiać - obracać (klawisz **R** - w prawo, **L** - w lewo) i tworzyć ich lustrzane odbicia (klawisz **H** - względem osi pionowej, **V** - poziomej). Właśnie w ten sposób ustawiono bramkę **AND** w przykładowym projekcie.
- Aby uniknąć zbędnego stosowania inwerterów, można użyć bramek z zanegowanymi wejściami, np. **AND 3B1** - to trzywejściowa bramka **AND** z jednym wejściem zanegowanym:



- Łączymy wybrane elementy albo poprzez wybór rodzaju linii na pasku zadań (klawisz **W**):  lub magistrali (klawisz **B**): . Wciśniętym lewym przyciskiem myszki wybieramy początek i koniec połączenia. Można także kliknąć na końcówce danego elementu, co powoduje automatyczne wywołanie opcji łączenia. Należy zwracać uwagę, czy na końcówce łącza jest różowa kropka - jeśli tak, to linia nie została dokładnie dociągnięta i nie ma połączenia (jest to bardzo częsty błąd, który objawia się m.in. pojawianiem się stanów nieustalonych podczas symulacji).
- Zakończenie nowej linii na już istniejącej powoduje ich połączenie (jeśli na skrzyżowanych liniach nie ma kropki, to nie są one połączone)

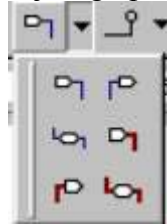
- Jeśli chcemy jakąś linią wprowadzić sygnał wejściowy, lub odczytać wyjściowy, konieczne jest użycie terminatorów lub końcówek **GND** - masa (klawisz **G**) i **VCC** - zasilanie (klawisz **P**). Dla zwykłych linii :

- wejścia - klawisz **I**
- wyjścia - klawisz **O**

Dla magistrali :

- wejścia - **Shift + I**
- wyjścia - **Shift + O**

Można też skorzystać z menu rozwijanego po naciśnięciu ikony terminatora :

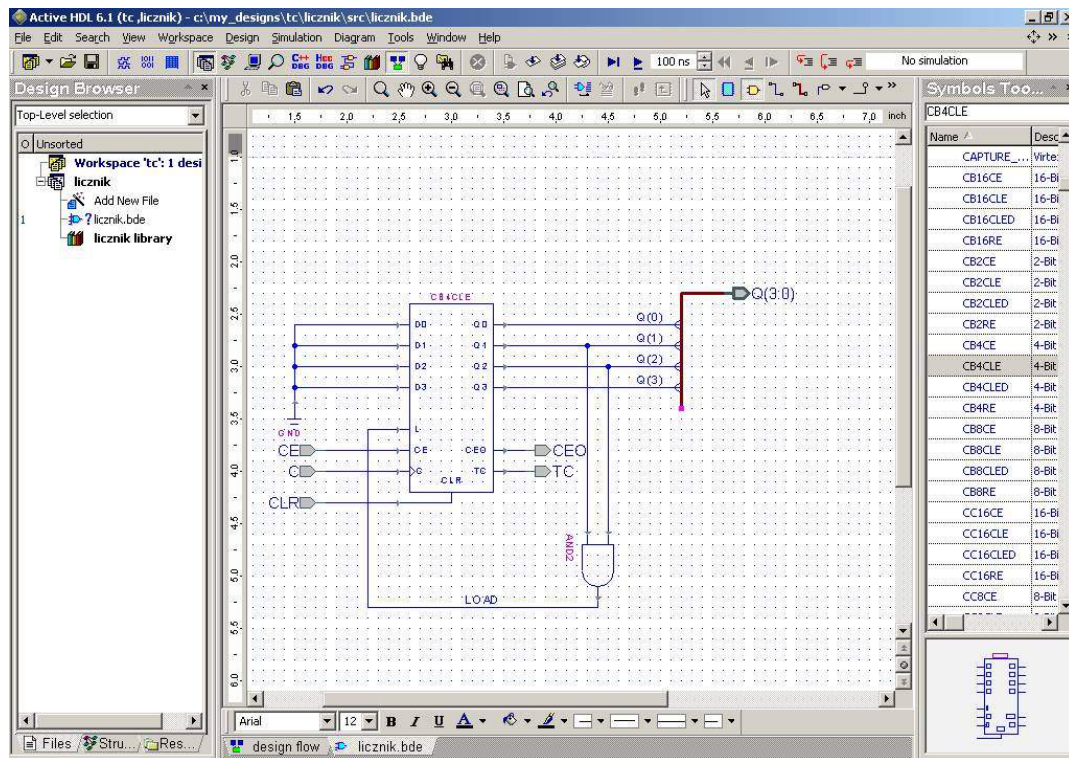


Sygnały wejściowe definiuje się podczas symulacji. Gdy korzysta się z masy i zasilania w sposób opisany powyżej, trzeba także zdefiniować ich wartość. Nie jest to konieczne jeśli korzysta się z końcówek **GND** i **VCC** z biblioteki, np. **Virtex**.

- W celu łatwej identyfikacji każde wejście i wyjście powinno posiadać nazwę (standardowo program nazywa je **Input/Output n**). Można ją ustawić dwukrotnie klikając na elemencie - pojawi się wtedy menu w którym ustawiać można różne parametry, m.in. nazwę. Innym sposobem jest dwukrotne kliknięcie na nazwie terminatora - wtedy jej edycja następuje bezpośrednio na schemacie.
- Istnieje również możliwość nadania osobnej nazwy każdemu połączeniu (w ten sam sposób jak w przypadku terminatorów) - w naszym projekcie nazwano linię **LOAD** podającą sygnał na wejście **L** licznika. Jest to konieczne w przypadku linii podpiętych do magistrali - ich nazwy muszą mieć postać **Nazwa(Numer)**, np. **O(1)**. Taką samą nazwę musi posiadać terminator magistrali, a zamiast numeru musi mieć podany zakres. W przykładowym projekcie do magistrali podpięte są linie **Q(0)**, **Q(1)**, **Q(2)**, **Q(3)**, a wyjście ma nazwę **Q(3:0)**. Zakres wyjścia można wpisać ręcznie, lub ustawić w menu, które pojawia się po dwukrotnym kliknięciu na wyjściu.

Uwaga!! Program rozróżnia małe i duże litery w nazewnictwie elementów – *q(1)* i *Q(1)* to dwa różne oznaczenia!!

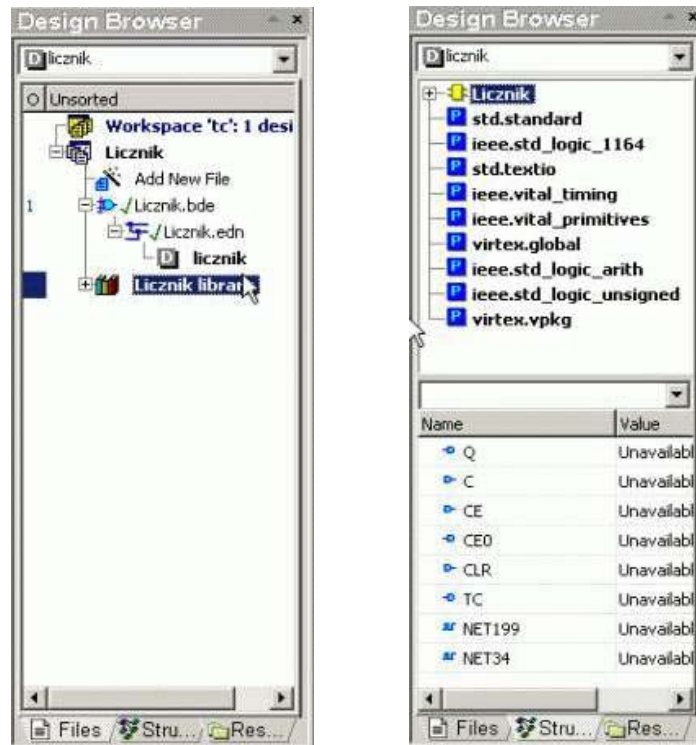
- Jeśli w projekcie wykorzystuje się wiele elementów, które posiadają wejścia o takich samych nazwach, np. wejścia zegarowe **C** przerzutników, zamiast rysowania wielu linii podpiętych do jednego wejścia można skorzystać z połączenia **Global Wire** - ikona: . Nazwanie takiego elementu oraz wejścia **C** powoduje, że wszystkie wejścia zegarowe **C** będą sterowane takim sygnałem, jaki poda się w symulacji na **Global Wire**. Zapewnia to przejrzystość projektu i eliminuje przypadkowe błędy takie jak połączenie jakiejś linii z sygnałem zegarowym.
- Powyższe informacje powinny swobodnie wystarczyć do stworzenia prostego, poprawnie działającego projektu, taki jak ten licznik :



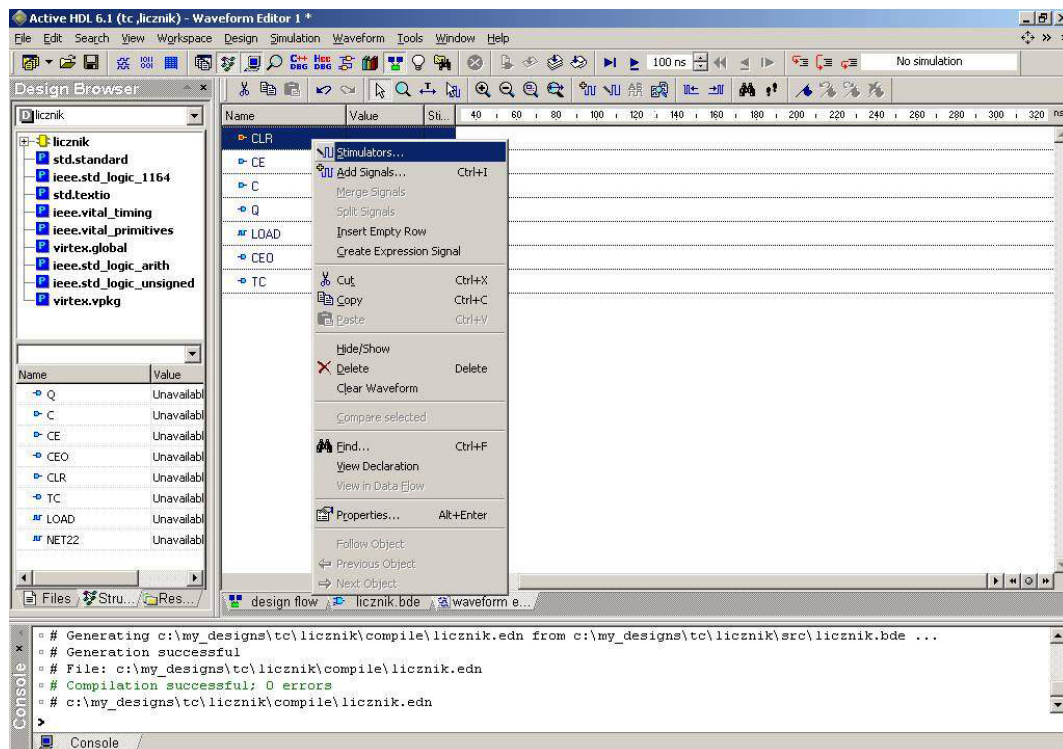
Symulacja i ćwiczenie

1. Przygotowanie symulacji:

- Jeżeli schemat został zakończony wówczas należy przeprowadzić kompilację. W tym celu z górnego paska zadań wybieramy **Design, Compile** (lub używamy przycisku **F11**). Jeśli w projekcie były jakieś błędy, program wypisze je w nowym oknie i poinformuje o nieudanej kompilacji w **Console**.
- Po kompilacji w oknie **Design Browser** widzimy całą strukturę projektu, dodatkowo pojawi się nasz plik z narysowanym schematem oraz struktura wejść i wyjść naszego układu (rysunek po prawej).



- W oknie **Design Browser** lub w menu **Design/Settings** wybieramy nasz projekt jako **Top Level**. Następnie należy otworzyć okienko z przebiegami **Waveform** i wybrać z menu **Simulation/Initialize Simulation**. W celu przeprowadzenia symulacji dodajemy sygnały, które chcemy zaobserwować poprzez przeniesienie ich z aktywnego okna powyżej (po prawej stronie).
- Teraz należy przyłożyć przebiegi do wejść - po naciśnięciu prawego klawisza na danym wejściu pojawi się rozwijane menu. Należy wybrać w nim opcję **Stimulators**.



Istnieje kilka rodzajów wymuszeń :

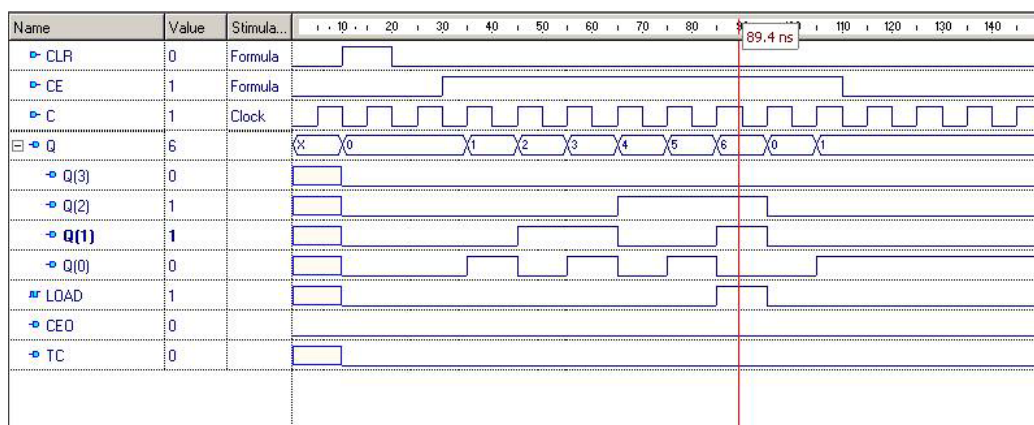
- **Clock** - zegar - podaje się częstotliwość i współczynnik wypełnienia. W naszej symulacji należy podać na wejście **C** sygnał o częstotliwości 100 MHz
- **Formula** - pozwala na stworzenie dowolnego wymuszenia. W naszym projekcie wymuszenie na wejściu **CLR** to 0 0 ns, 1 10 ns, 0 20 ns, a na wejściu **CE** to 0 0 ns, 1 30 ns, 0 110 ns
- **Value** - przypisuje stałą wartość wymuszenia. Jeśli w projekcie wykorzystano standardowe elementy **GND** i **VCC**, to należy przypisać im odpowiednio wartości 0 i 1
- **Counter** - zliczanie w dół lub w górę (można określić kod w jakim działa licznik) - stosowany w przypadku podawania wymuszenia na magistralę, np. do wybierania kolejnych adresów w pamięci (będzie użyteczny w przyszłości ☺)
- **Hotkey** - przypisanie klawisza, który pozwala na zmianę wymuszenia w trakcie symulacji. Istotne jest tutaj, aby czas symulacji (o tym później) był krótki, najlepiej równy jednemu okresowi zegara dla układów synchronicznych, w przeciwnym razie zmiany będą możliwe tylko co jeden okres symulacji. Żeby sprawdzić własnoręcznie działanie tego wymuszenia można np. ustawić wymuszenie jednego z wejść **CLR** lub **CE** na **Hotkey**

Na wejściach i wyjściach może występować kilka stanów :

- **"1"** – stan wysoki
- **"0"** – stan niski
- **"Z"** – stan wysokiej impedancji
- **"X"** – stan nieokreślony
- **"U"** – wartość niezdefiniowana

2. Symulacja:

- Po ustawieniu sygnałów należy ustawić czas symulacji i uruchomić ją (klawisz **F5** lub ikona **Run For**)  i to właściwie jest koniec pracy. Czas można ustawić tak, aby wszystkie przejścia wykonały się w jednej symulacji, lub tak, aby pełna symulacja składała się z kilku etapów, co pozwala na dokładniejsze śledzenie zmian i sterowanie **Hotkey**.
- Po przeprowadzeniu symulacji powinniśmy otrzymać taki wynik :



- Ustawiając znacznik na wybranej chwili czasowej możemy w kolumnie *Value* odczytać jakie były wtedy stany poszczególnych wejść i wyjść.

3. Omówienie projektu, czyli co tak właściwie symulowaliśmy?

W naszym projekcie użyliśmy licznika *modulo16* i bramki *AND2* z biblioteki *Virtex* do zbudowania licznika *modulo7*. Początkowe ustawienie "0" zapewnia pojedynczy impuls na wejściu *CLR* (asynchroniczny reset), jednak licznik nie rozpocznie zliczania, dopóki mu się na to nie zezwoli, ustawiając stan "1" na wejściu *CE* – *Clock Enable*. Po spełnieniu tych warunków licznik zaczyna zliczać w górę i w normalnych warunkach powinien dojść do wartości "15", jednak za sprawą bramki *AND* tak się nie dzieje. Jak wiadomo "1" na wyjściu bramki *AND* pojawia się gdy na wszystkich jej wejściach mamy stan "1". W naszym przypadku wymaga to pojawienia się "1" na liniach *Q(1)* i *Q(2)*. Wyjście *Q(0)* to *LSB*, a *Q(3)* to *MSB*, czyli jedynki na liniach *Q(1)* i *Q(2)* pojawią się gdy licznik doliczy do "6" (110 binarnie). Pojawienie się "1" na linii *LOAD* powoduje załadowanie do licznika wartości podanej aktualnie na wejścia *D*. W naszym przypadku jest to "0", ponieważ wszystkie wejścia są podpięte do masy. Tak więc po osiągnięciu wartości "6" licznik zaczyna zliczać od początku – czyli zbudowaliśmy licznik *modulo7*.

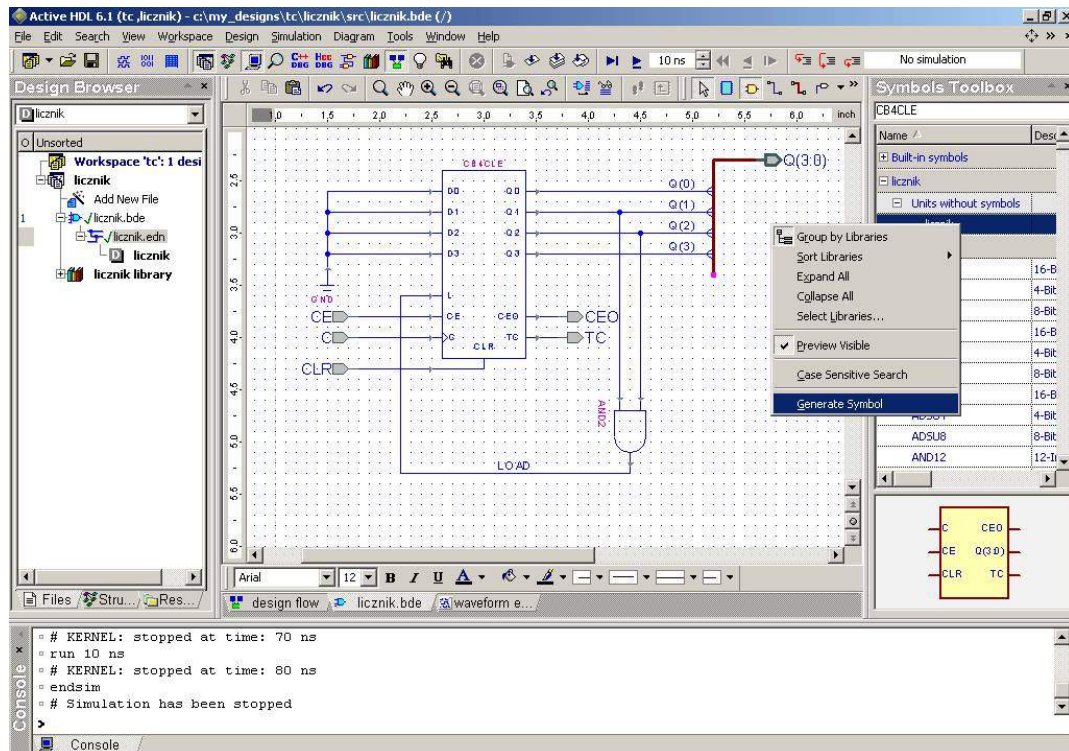
4. Małe ćwiczenie

Jak zauważyliśmy wcześniej licznik biblioteczny *modulo16* nie jest w pełni wykorzystany. Nie wiemy także w jaki sposób działają wyjścia *TC* i *CEO* – cały czas jest na nich stan "0". Aby przekonać się jak one działają, potrzebna będzie mała modyfikacja naszego projektu. Aby licznik mógł zliczać do "15" trzeba usunąć bramkę *AND* i podłączyć wejście *L* do masy. Symulacja powinna trwać na tyle długo, aby licznik doliczył do końca – co wtedy pojawia się na wyjściach *TC* i *CEO*?? A co się stanie, gdy po dojściu do stanu "15" stan *CE* zmieni się na "0"?? Odpowiedź na te pytania powinna wyjaśnić działanie obu wyjść. Do czego można wykorzystać te sygnały?? A jak będzie się zachowywał licznik gdy do wejścia *L* podepnie się sygnał z wyjścia *TC* lub *CEO*??

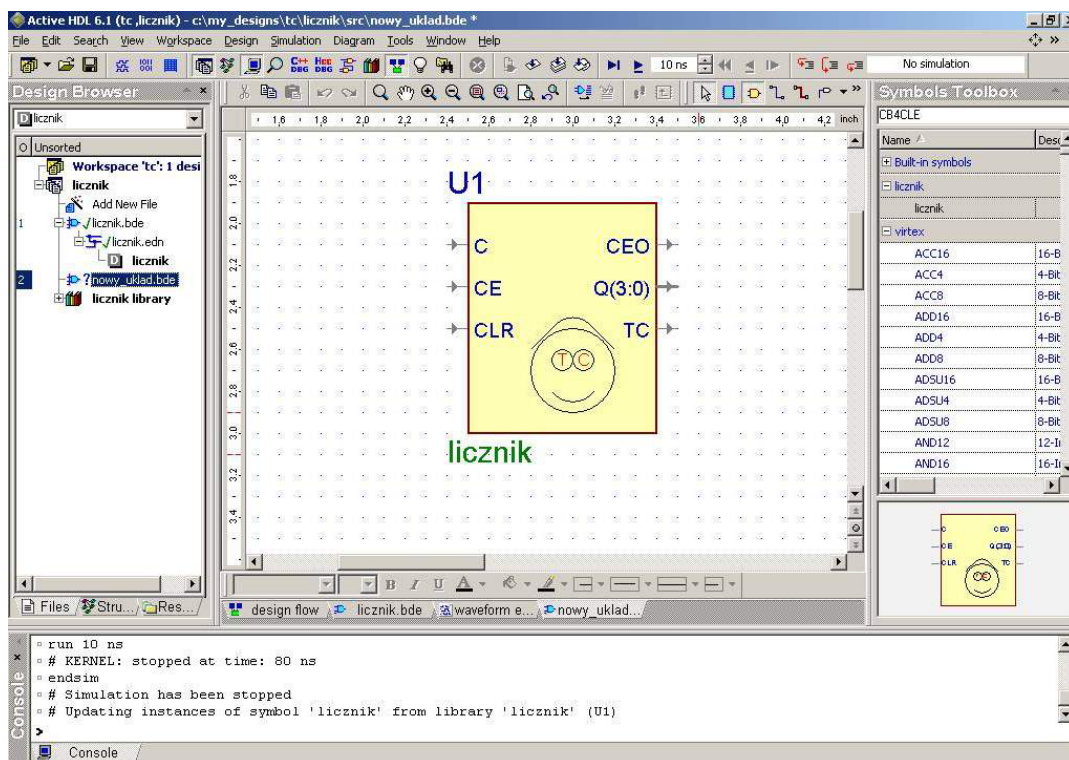
Uwagi i wskazówki

1. Omawiany projekt jest bardzo prosty. Jednak na ogół układy logiczne są znacznie bardziej skomplikowane. Chcąc zbudować układ oparty np. na dwóch rejestrach 7-bitowych potrzebne jest 14 przerzutników. Jeśli rejestry te mają mieć możliwość załadowania danych, a następnie ich przesuwania, konieczne będą bramki realizujące odpowiednie funkcje logiczne. Aby dodać dane zapisane w rejestrach konieczne jest zbudowanie układu sumacyjnego i nagle okazuje się, że prosty projekt jest zbudowany z kilkudziesięciu elementów i plątaniny połączeń. Zadanie można sobie znacznie ułatwić konstruując odpowiednie bloczki. Procedura jest bardzo prosta :
 - tworzymy nowy plik *BDE* wewnątrz naszego projektu
 - budujemy odpowiedni układ pamiętając o dobrym nazewnictwie końcówek
 - kompilujemy bloczek
 - przechodzimy do głównego pliku *BDE* i otwieramy okienko *Symbol Toolbox* (klawisz *S*) – oprócz normalnych bibliotek powinien pojawić się pasek z nazwą projektu z rozwijalnym menu
 - rozwijamy menu projektu i zawarte w nim menu *Units Without Symbols*

- klikamy prawym klawiszem na nazwie skompilowanego właśnie pliku i wybieramy opcję *Generate Symbol* i... to tyle!!



- Można jeszcze zmienić wygląd elementu wybierając z tego samego menu opcję *Edit Symbol*.



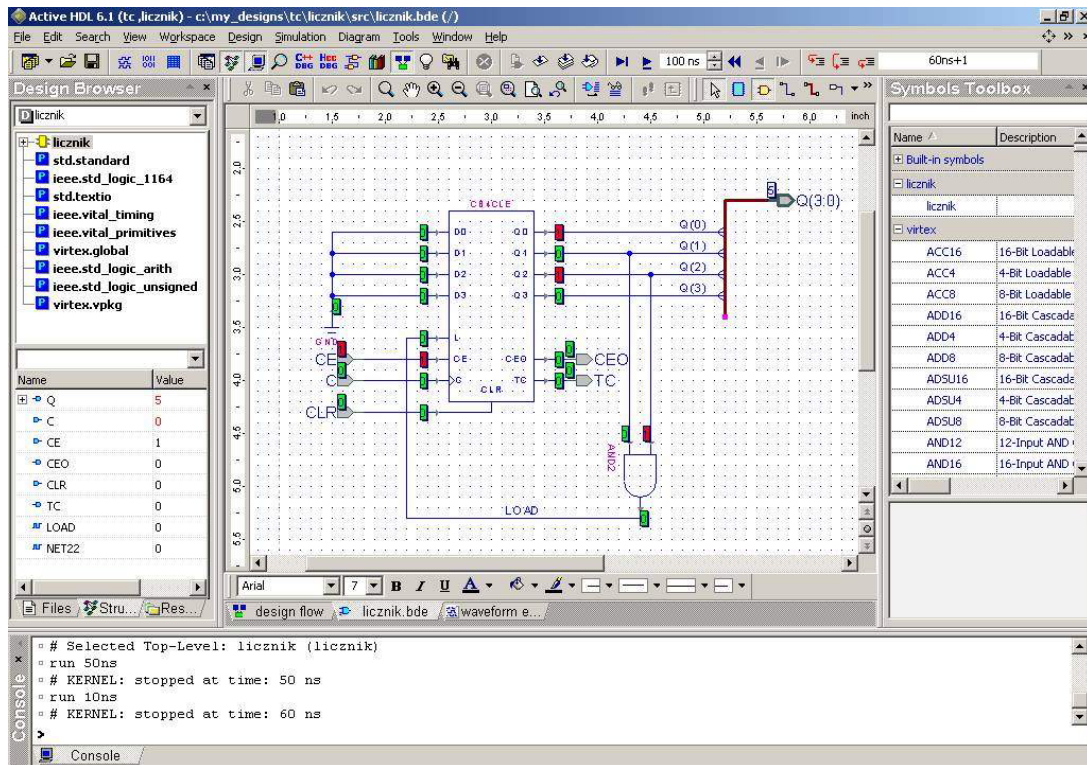
- wewnątrz elementu można obejrzeć dwukrotnie klikając na jego symbolu

- Stworzony element można także wykorzystywać w innych projektach wewnątrz danego *Workspace'a* ale wtedy trzeba dodać do podstawowych bibliotek projekt do którego należy dany element (*Select Libraries...*)
2. W niektórych projektach warto skorzystać z możliwości sterowania elementami, które nie posiadają odpowiednich wejść, np. do resetowania przerzutników typu **D**, nie posiadających wejść **Clear** lub **Reset**. Jest to przydatne, gdyż często użycie elementu **FD** ("goły" przerzutnik **D**) powoduje powstawanie problemów – układ po prostu nie działa. Zmienne sterujące elementami można znaleźć po skompilowaniu projektu w okienku *Design Browser* w punkcie *virtex.global*. Zmienna odpowiadająca za reset to **GSR**:



Symulacja którą wcześniej przeprowadzaliśmy to tzw. symulacja funkcjonalna (*Functional Simulation* w *Design Flow*). Jest ona tylko jednym z etapów projektowania. Po jej wykonaniu należy dokonać (automatycznej) implementacji projektu (*Implementation*) – przetworzenia teoretycznego projektu na projekt zbudowany z rzeczywistych elementów danej biblioteki (nie działa w wersji demonstracyjnej programu). Jeśli implementacja zakończyła się sukcesem, można przeprowadzić symulację czasową (*Timing Simulation*) w której widać rzeczywiste opóźnienia czasowe w układzie.

3. Aby zbadać dokładnie stany układu w kolejnych chwilach czasowych po uruchomieniu symulacji, można przejść do okna schematu – przy każdym wejściu i wyjściu będzie podany jego aktualny stan :



4. Na koniec przypomnienie kilku zasad projektowania:

- dbać o czytelność projektu – ustawiać elementy tak, aby połączenia były jak najkrótsze i jak najmniej poplątane
- korzystać z magistral i z **Global Wire**
- nie mieszać elementów z różnych bibliotek
- dbać o nazewnictwo końcówek i połączeń
- korzystać z możliwości tworzenia bloków