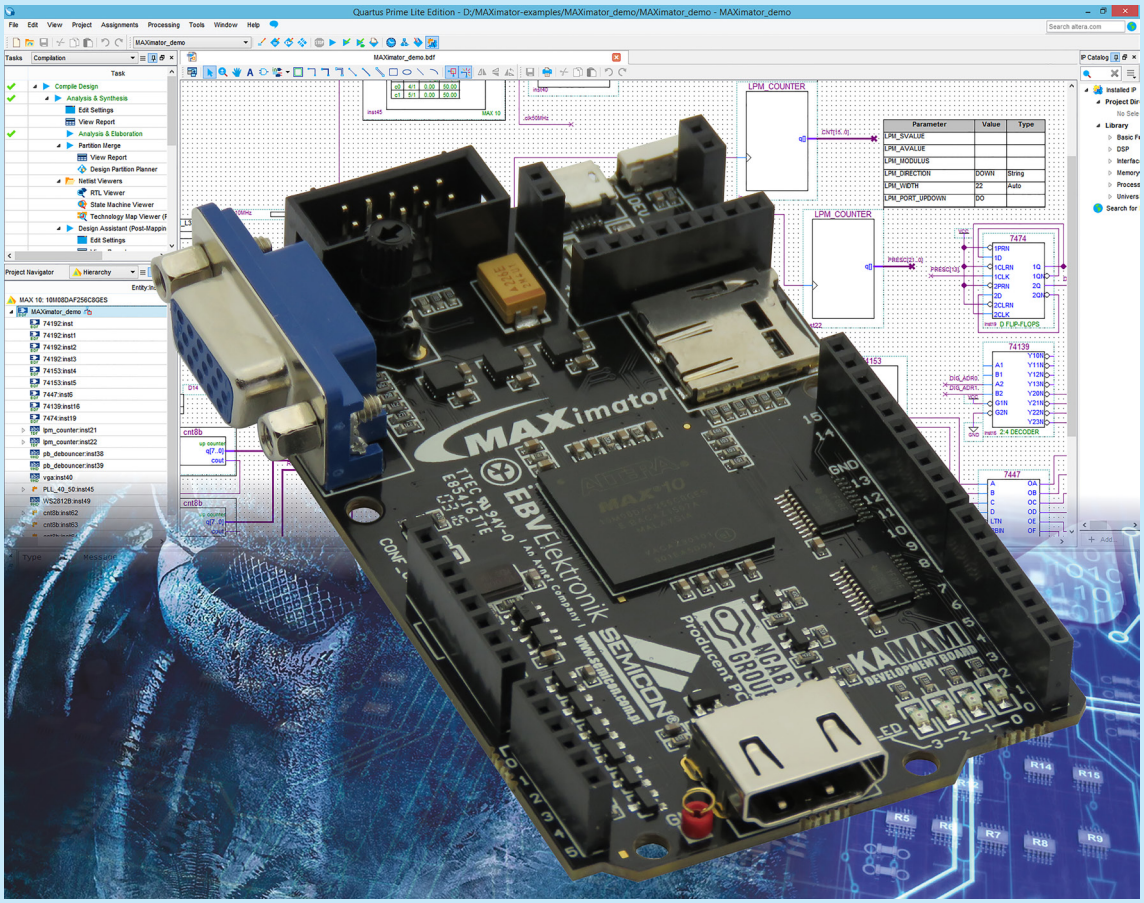


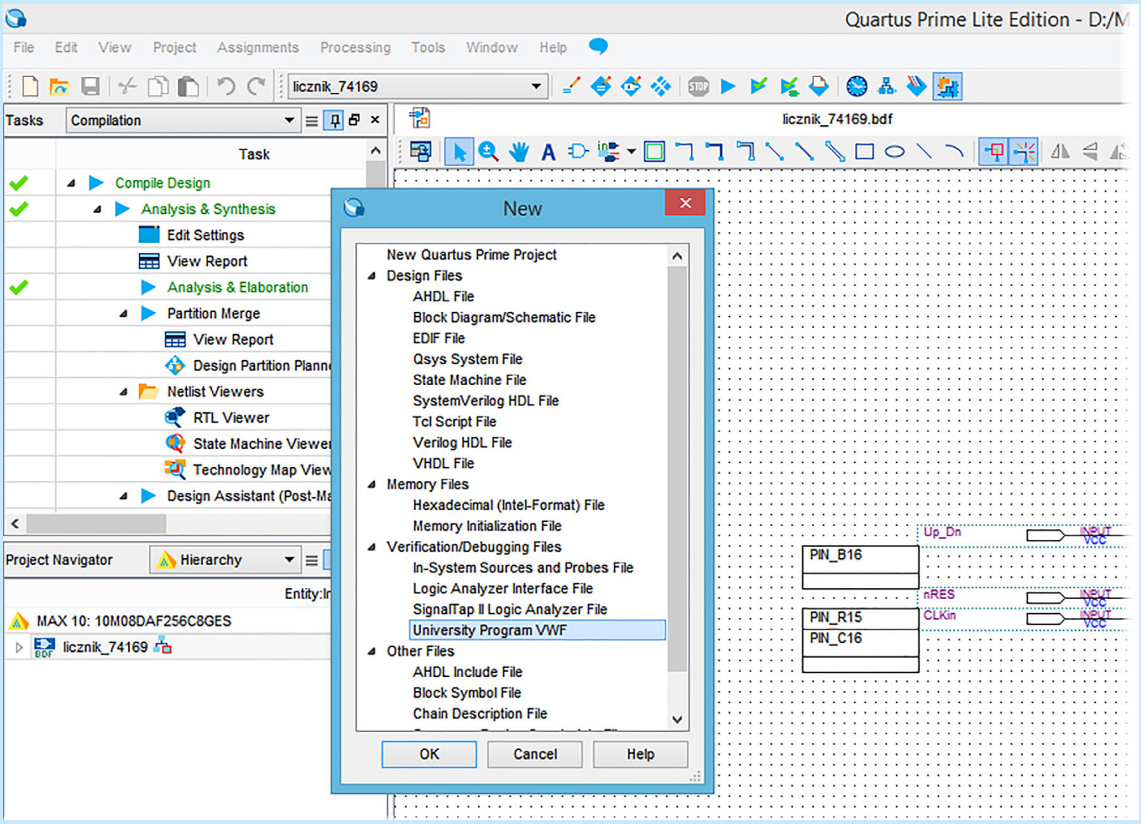
Pierwsze kroki z FPGA (3)

Szkoła MAXimatora – testowanie funkcjonalne i weryfikacja działania projektu w środowisku Quartus Prime z wykorzystaniem symulatora ModelSIM.

W tej części artykułu przedstawimy krok-po-kroku symulację projektu licznika, którego przygotowanie opisaliśmy miesiąc temu. Do symulacji użyjemy bezpłatnego symulatora ModelSIM oraz wbudowanego w Quartus Prime graficznego edytora przebiegów.

Środowisko projektowe Quartus Prime zostało wyposażone przez producenta w kilka narzędzi umożliwiających symulowanie funkcjonalne projektów implementowanych w FPGA. W artykule pokażemy najprostszy z nich, wykorzystujący symulator wbudowany w środowisko. Jego właściwości użytkowe są wystarczające do symulowania projektów implementowanych w bezpłatnej wersji programu Quartus Prime, bazuje on na specjalnej, bezpłatnej wersji zewnętrznego symulatora ModelSIM firmy Mentor Graphics. Za edycję i prezentację przebiegów będących wynikiem symulacji odpowiada wygodny w użyciu *Waveform Editor*.

Symulacja projektu wymaga przygotowania dla aktualnie otwartego projektu pliku o rozszerzeniu *vwf*, co wymaga wybrania w oknie *File* → *New*, w sekcji *Verification/Debugging Files*, opcji *University Program VWF* (**rysunek 1**). Spowoduje to otwarcie okna edytora przebiegów *Waveform Editor*, które pokazano na **rysunku 2**. Jak widać jest ono puste, musimy wprowadzić do opisu symulacji wszystkie linie wejściowe i wyjściowe oraz sygnały (węzły) wewnętrzne, których działanie chcemy uwzględnić w testach. W tym celu stajemy myszką w lewej części okna edytora i klikamy prawym przyciskiem myszy, co spowoduje wyświetlenie menu kontekstowego (**rysunek 3**), z którego wybieramy opcję *Insert Node or Bus…*, co spowoduje wyświetlenie okna pokazanego na **rysunku 4**. Żeby wyświetlić wszystkie interesujące nas sygnały, linie wejściowe i wyjściowe, należy przycisnąć przycisk *Node Finder…*, w wyniku czego zostanie wyświetlone okno jak na **rysunku 5**. W zależności od tego, jakiego rodzaju sygnały nas interesują, warto użyć preselektora (filtru), który znajduje się w górnej prawej części wyświetlonego okna (**rysunek 6**), przy czym trzeba pamiętać,



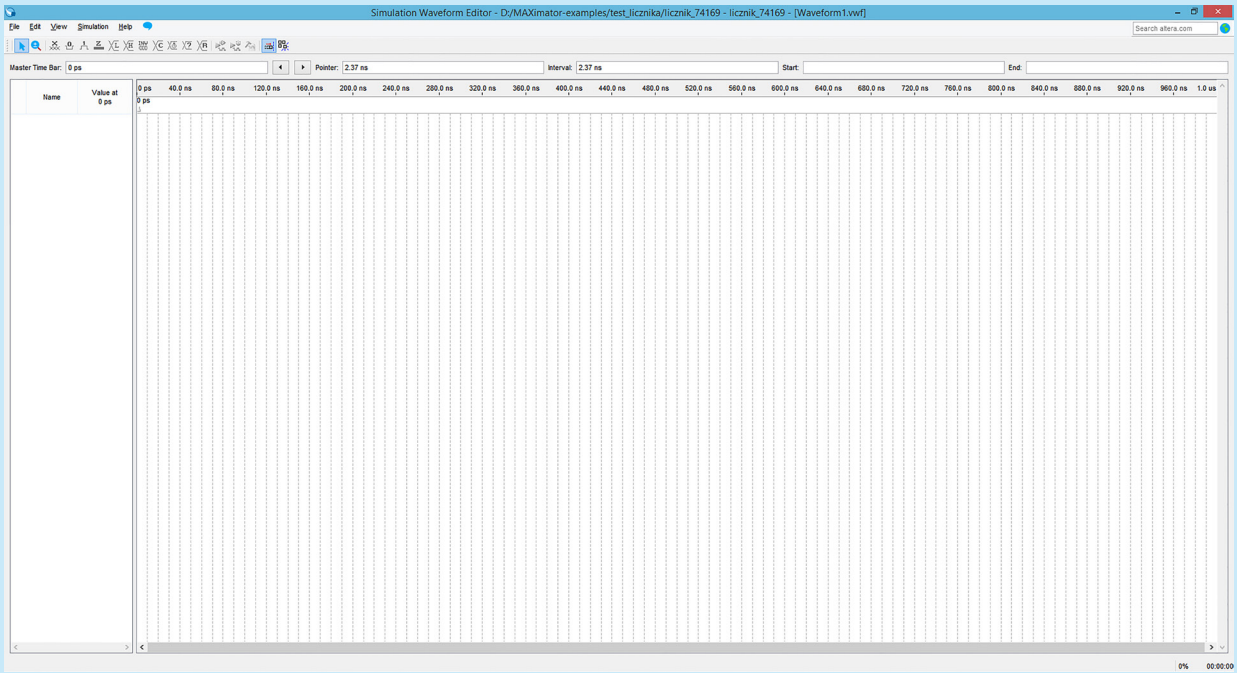
Rysunek 1. Symulacja projektu wymaga przygotowania dla aktualnie otwartego projektu pliku o rozszerzeniu *vwf*

że po ustaleniu trybu filtrowania trzeba każdorazowo nacisnąć przycisk *List*, który odświeża listę wyświetlonych sygnałów.

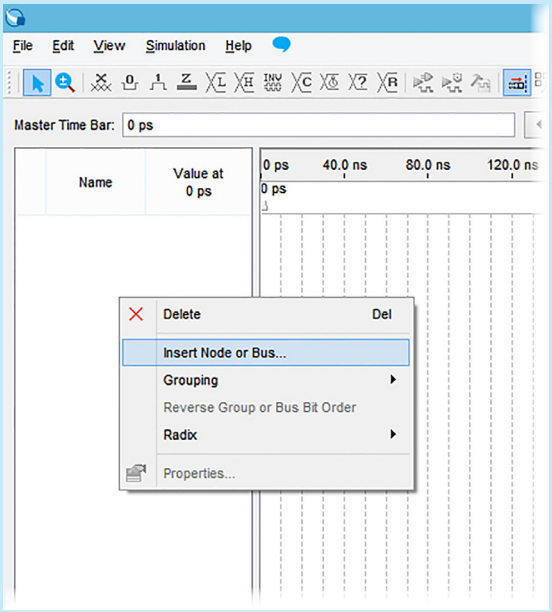
W naszym przypadku skupimy się na analizie zmian stanów na liniach wyjściowych w zależności do zmian stanów na liniach wejściowych, więc filtr powinien być ustawiony na *Pins: all*. Listę znalezionych przez symulator linii wejściowych i wyjściowych widać na rys. 5, warto ją porównać z naszym projektem (**rysunek 7**) – jak widać, w obydwu przypadkach nazwy linii wejściowych i wyjściowych są identyczne.

Po wybraniu linii i/lub sygnałów, które będą uwzględniane podczas symulacji (w naszym przypadku będą to wszystkie linie) przenosimy je do listy *Selected Nodes* (**rysunek 8**). Wybór zatwierdzamy przyciskiem *OK*, po zatwierdzeniu w kolejnym oknie także za pomocą *OK*, wybrane sygnały zostaną wyświetlone w edytorze przebiegów jak pokazano na **rysunku 9**. Liniom wejściowym domyślnie są przypisane stany logicznego „0”, a wyjściom stany nieustalone.

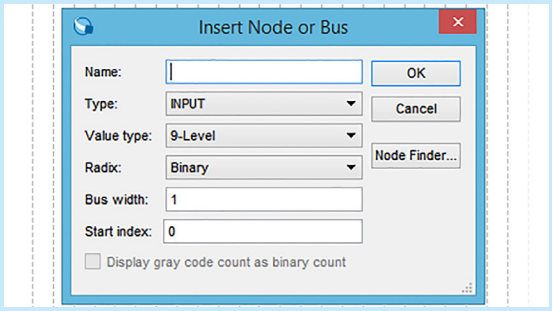
Teraz musimy przypisać stany linii wejściowych, na które będziemy badać reakcje wyjść zaprojektowanego



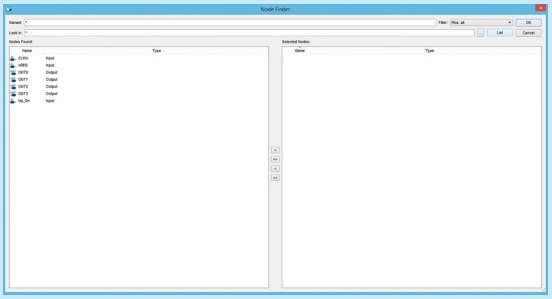
Rysunek 2. Okno edytora przebiegów *Waveform Editor*



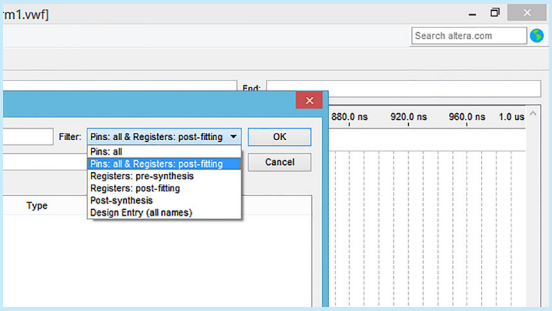
Rysunek 3. Wyświetlenie menu kontekstowego



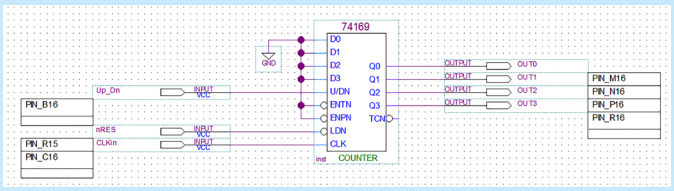
Rysunek 4. Okno *Insert Node or Bus*



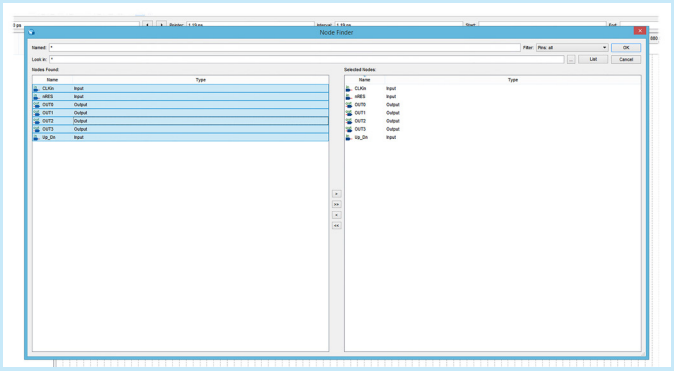
Rysunek 5. Okno *Node Finder*



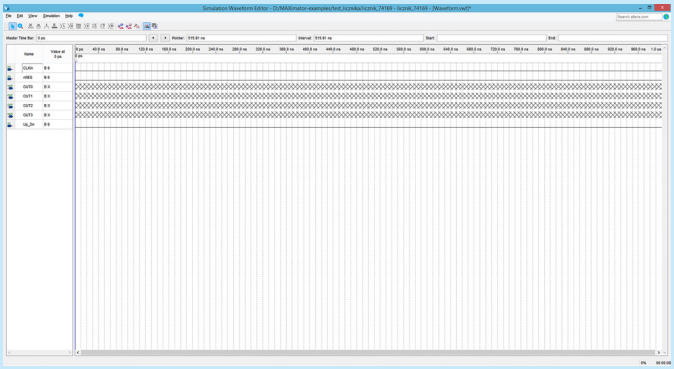
Rysunek 6. Okno filtru (preselektora)



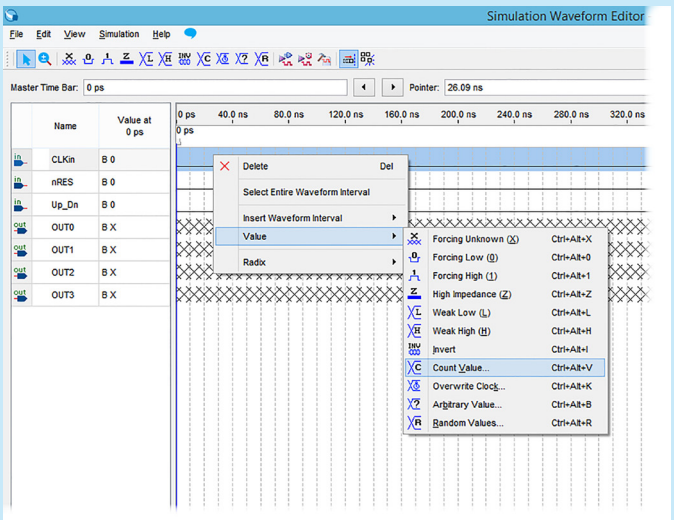
Rysunek 7. Schemat projektu



Rysunek 8. Przeniesienie sygnałów do *Selected Nodes*

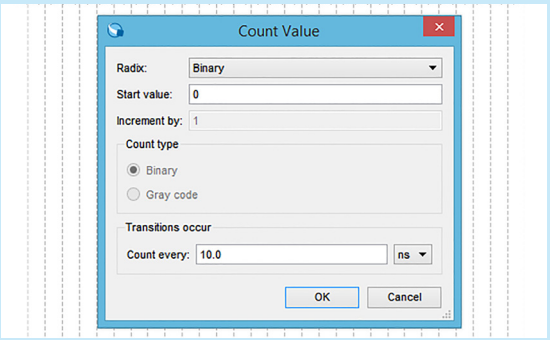


Rysunek 9. Wyświetlenie wybranych sygnałów w edytorze przebiegów

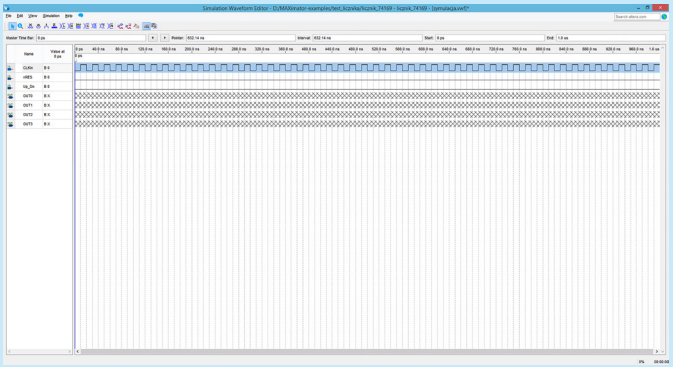


Rysunek 10. Wybór opcji *Value* ➔ *Count Value* z menu kontekstowego

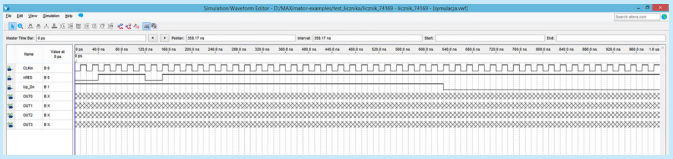
układu. Zaczniemy od utworzenia na wejściu CLKin sygnału zegarowego, co wymaga ustawienia kursora myszki w dowolnym miejscu edytowanego przebiegu i naciśnięcia prawego przycisku myszki. Z wyświetlonego menu kontekstowego wybieramy opcje *Value* → *Count Value* (**rysunek 10**), co spowoduje wyświetlenie okna edycji przebiegu zegarowego, które pokazano na **rysunku 11**. W pozycji *Transitions Occur* można zdefiniować odstępy czasowe pomiędzy zboczami sygnału zegarowego (dla wartości 10 ns okres przebiegu wynosi 20 ns).W wyniku opisanego ciągu operacji uzyskaliśmy przebieg jak pokazano na **rysunku 12**. W podobny sposób postępujemy z pozostałymi sygnałami wejściowymi, na **rysunku 13** pokazano przykładowe definicje przebiegów.



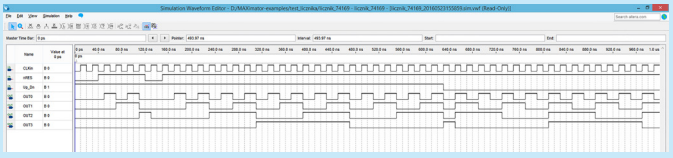
Rysunek 11. Okno edycji przebiegu zegarowego



Rysunek 12. Wyświetlenie symulowanego przebiegu



Rysunek 13. Przykładowe definicje przebiegów



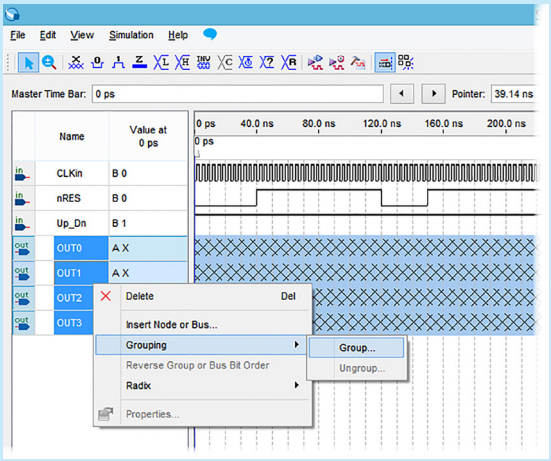
Rysunek 14. Wynik symulacji funkcjonalnej weryfikowanego układu

Symulator wbudowany w Quartus Prime umożliwia wykonanie dwóch rodzajów symulacji:

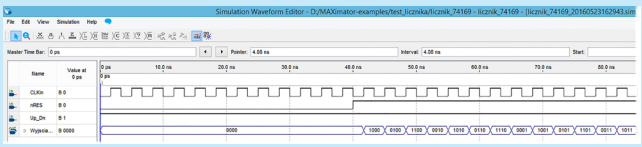
– Funkcjonalnej, podczas której brane są pod uwagę wyłącznie zależności logiczne pomiędzy sygnałami, bez uwzględnienia wpływu czasów ich propagacji na działanie układu.

– Czasowej, która jest bliższa realnym układom, bowiem oprócz zależności logicznych brane są pod uwagę także zależności czasowe pomiędzy sygnałami. Na **rysunku 14** pokazano wynik symulacji funkcjonalnej weryfikowanego układu.

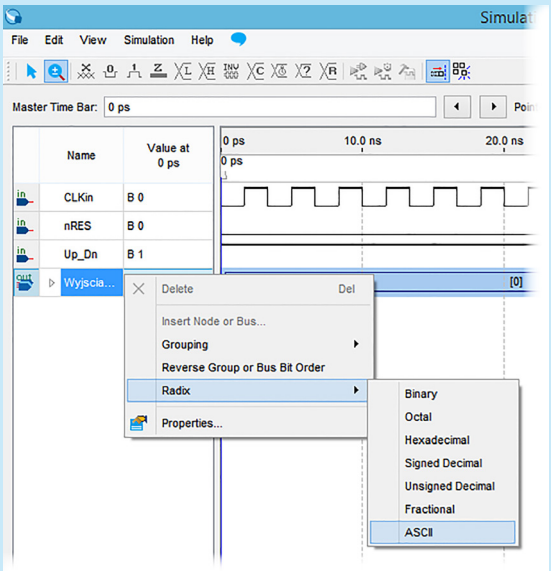
Edytor przebiegów wyposażono w kilka mechanizmów, które ułatwiają przygotowanie symulacji projektów o większej niż w przykładzie liczbie analizowanych wejść i wyjść. Można na przykład grupować wybrane



Rysunek 15. Grupowanie wybranych linii w magistrale



Rysunek 16. Wyświetlenie przebiegu na zgrupowanych liniach



Rysunek 17. Zmiana formatu wyświetlanych wartości

linie w magistrale (**rysunki 15** i **16**), można także zmieniać format wyświetlanych wartości w zależności od indywidualnych potrzeb (**rysunek 17**).

W ten sposób przeszliśmy kompletną ścieżkę implementacji prostego projektu w FPGA. W kolejnym wydaniu EP skupimy się na pokazaniu zaawansowanych mechanizmów weryfikacji projektów implementowanych w FPGA, koncepcyjnie zbliżonych do debugowania programów uruchamianych na mikrokontrolerach.

Piotr Zbysiński, EP