

Laboratorium #II-4. Wykorzystanie pamięci ROM do rysowania grafik na ekranie VGA.

- Obsługa pamięci ROM.
- Interakcja myszy z obiektem na ekranie.

Używane elementy: Basys3

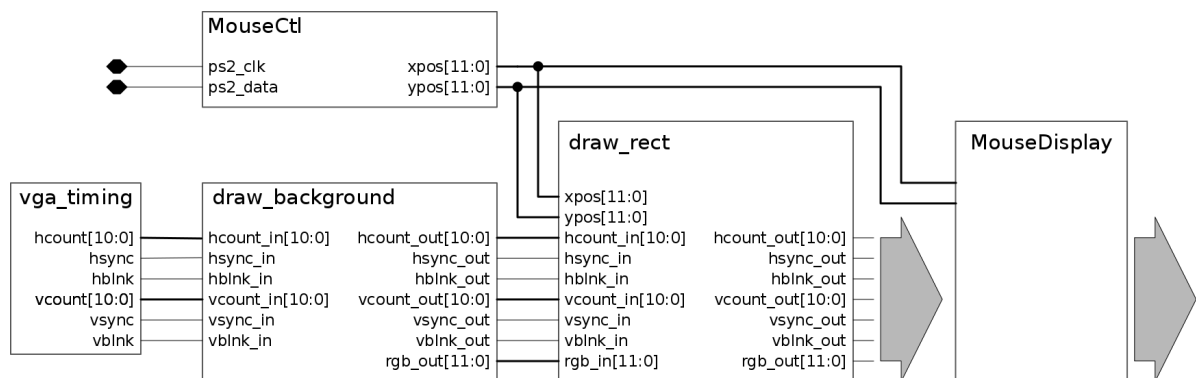
Temat ćwiczenia.

Zadanie do wykonania w ramach ćwiczenia to opracowanie programu, który będzie na ekranie wyświetlał prostokąt z graficzną zawartością, poruszany myszką. Po naciśnięciu przycisku myszy prostokąt ma opadać grawitacyjnie na ekranie, a potem odbić się z podanym współczynnikiem odbicia.

W poprzednim ćwiczeniu zaprojektowałeś układ rysujący na ekranie prostokąt poruszany myszką. W tym ćwiczeniu rozbudujesz opracowany wcześniej układ tak, aby umożliwić wyświetlanie graficznego obrazu wewnątrz prostokąta oraz reakcję na przycisk myszy.

Przebieg ćwiczenia

W poprzednim ćwiczeniu opracowałeś układ, który wyświetlał na ekranie prostokąt na podanym tle poruszany myszką. Uproszczony Schemat tego układu wyglądał następująco:



W tym ćwiczeniu dodamy do schematu jeszcze dwa moduły:

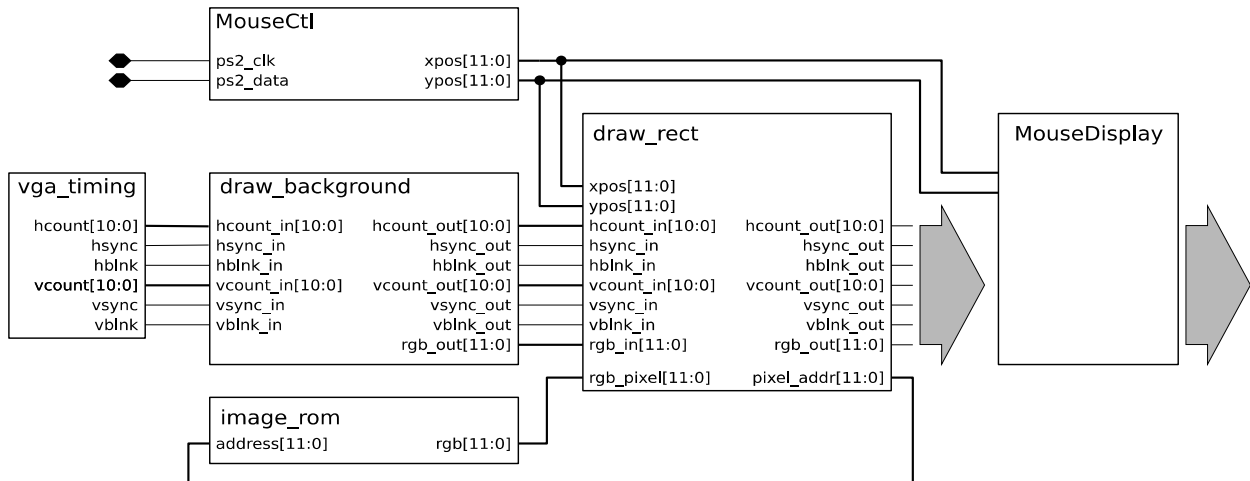
- moduł pamięci ROM zawierający obrazek graficzny do wyświetlenia na ekranie,
- dodatkowy moduł kontrolujący położenie prostokąta.

1. Dodanie pamięci ROM.

Zrób kopię zapasową projektu przed wprowadzaniem modyfikacji, jeżeli takiej nie posiadasz.

1.1. Moduł pamięci ROM znajduje się w pliku **image_rom.v**, dostępnym na UPEL w ramach ćwiczenia. Moduł zawiera mapę bitową RGB obrazka o wymiarach 48 x 64 pikseli. Dołącz ten moduł do projektu i przejrzyj jego zawartość. Na początku pliku, w komentarzach, znajdziesz wszystkie informacje potrzebne do podłączenia modułu. Zwróć uwagę, że moduł działa synchronicznie, tzn. zwraca wartość koloru dla wybranego piksela po narastającym zboczu zegara.

1.2. Dodaj nowe porty modułu **draw_rect** i podłącz moduł **image_rom** zgodnie ze schematem poniżej.



1.3. Zmodyfikuj zawartość modułu **draw_rect** tak, aby moduł wyświetlał prostokąt o wymiarze 48 x 64 pikseli z zawartością graficzną. Zwróć uwagę, że w celu utrzymania poprawnej synchronizacji musisz zapewnić opóźnienie sygnałów **hcount**, **hsync**, ... etc. o dwa takty zegara (lub więcej, w zależności od sposobu generacji adresu **pixel_addr**).

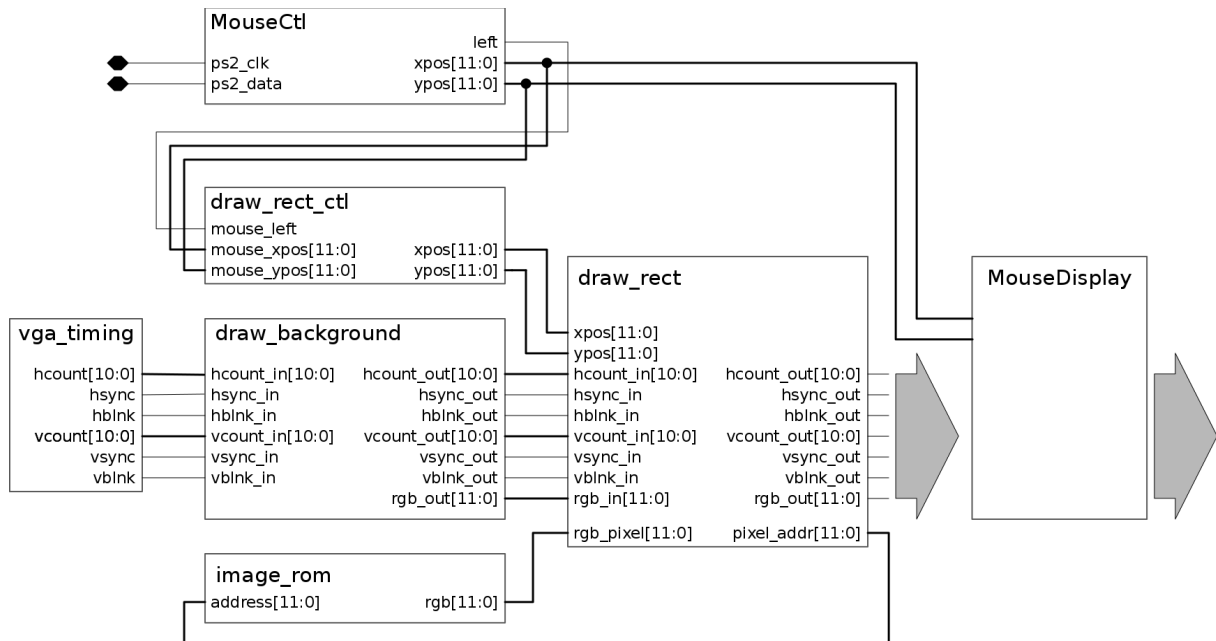
1.4 Skompiluj i uruchom na płytce program. Powinien on wyświetlać zdefiniowany przez Ciebie prostokąt z zawartością graficzną poruszający się po ekranie zgodnie z ruchami myszy.

1.5. Sprawdź ostrzeżenia w zakładce „Messages” - czy wszystkie rozumiesz?

2. Interakcja myszy z obiektem na ekranie.

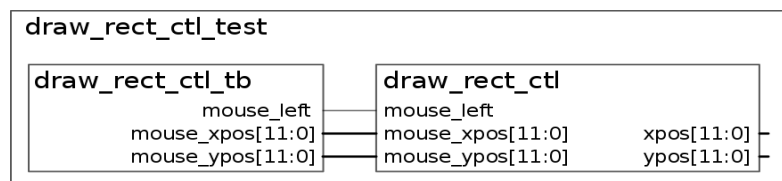
Zrób kopię zapasową projektu przed wprowadzaniem kolejnych modyfikacji, jeżeli takiej nie posiadasz.

Celem tego kroku uzyskanie efektu „spadania” prostokąta po ekranie po naciśnięciu lewego przycisku myszy. Aby uzyskać opisany efekt dodasz do projektu kolejny moduł **draw_rect_ctl**, którego zadaniem będzie sterowanie zachowaniem prostokąta na ekranie. Moduł będzie przejmował sygnały z myszy i na ich podstawie generował pozycję prostokąta. Docelowy schemat systemu jest przedstawiony poniżej. Przyjrzyj mu się, ale nie konstruuuj go jeszcze.



2.1. W pierwszym kroku zajmiesz się samym modulem **draw_rect_ctl**. Utwórz ten moduł oraz dwa moduły do jego testowania:

- moduł **draw_rect_ctl_tb** – do generowania przebiegów testowych (symulator myszy),
- moduł **draw_rect_ctl_test** – definiujący połączenia pomiędzy dwoma wcześniej wymienionymi.



2.2. Napisz zawartość modułów testowych oraz modułu **draw_rect_ctl** i przeprowadź symulację. Symulację należy przeprowadzić dla sygnału zegara o częstotliwości jaka będzie użyta w docelowym systemie.

Testbench powinien zapewniać:

- reset,
- ustawienie pozycji myszy w górze ekranu przy nie wciśniętym przycisku myszy (**mouse_left = 0**),
- wciśnięcie przycisku myszy (**mouse_leftb = 1**),
- czas na obserwację zmian po wciśnięciu przycisku myszy.

Moduł **draw_rect_ctl** powinien zapewniać:

- przepisywanie na wyjście współrzędnych myszy przy nie wciśniętym lewym przycisku myszy,
- po wciśnięciu przycisku myszy współrzędne na wyjściu modułu powinny się zmieniać tak, aby sterowany prostokąt grawitacyjnie spadał na dół ekranu (i zatrzymał się na dole). W pierwszym podejściu można przyjąć stałą prędkość opadania. Należy tak ustalić przyspieszenie, jakie ma ciało spadające bez oporu w próżni (weź pod uwagę rozmiar ekranu). Uwaga: może się okazać, że będziesz potrzebował użyć liczb ułamkowych; najprostszym rozwiązaniem, jest tutaj stosowanie liczb stałoprzecinkowych.
- opcjonalnie: można dodać odbijanie się obiektu od dna ekranu z wybranym współczynnikiem, np. 0.8 (prędkości lub energii).

Obserwuj wyjścia **xpos[11:0]** i **ypos[11:0]** modułu **draw_rect_ctl** w symulacji. W wyniku symulacji powinieneś być w stanie odpowiedzieć na pytanie, ile czasu zajmuje opadanie prostokąta na dno ekranu.

Możesz również wykorzystać komendę **\$strobe** i wypisać na ekran czas i **ypos** tak, aby możliwe było załadowanie tych danych do arkusza kalkulacyjnego i wyświetlenie wykresu dla tych zmiennych.

2.3. Dodaj moduł **draw_rect_ctl** do głównego projektu zgodnie z wcześniej podanym schematem. Skompiluj i uruchom na płytce program. Powinien on wyświetlać zdefiniowany przez Ciebie prostokąt reagujący na mysz zgodnie ze specyfikacją.

2.4. Sprawdź ostrzeżenia w zakładce „Messages” - czy wszystkie rozumiesz?

Wyniki ćwiczenia

Jako wynik ćwiczenia należy:

- załadować projekt na UPEL (bez katalogu symulacyjnego i implementacyjnego).

ZA BRAK ZAŁADOWANEGO PROJEKTU NA UPEL OTRZYMUJEMY -1 pkt. DO OCENY.

- zaprezentować działanie programu na następnych zajęciach laboratoryjnych i wyjaśnić słownie zasadę działania.

UWAGA: Kod projektowanego modułu powinien być napisany zgodnie z zasadami opisanymi w pliku „Zasady pisanie kodu w języku Verilog”, dostępnego na UPEL. Będzie to oceniane!

Wyniki będą ocenione bezpośrednio na kolejnych zajęciach.