Laboratorium #II-3 Poruszanie obiektami na ekranie VGA

- Generacja zsynchronizowanych sygnałów zegarowych.
- Podłączenie modułu napisanego w języku VHDL do projektu w języku SystemVerilog.
- Obsługa myszy PS/2.

Używane elementy: Basys3, mysz

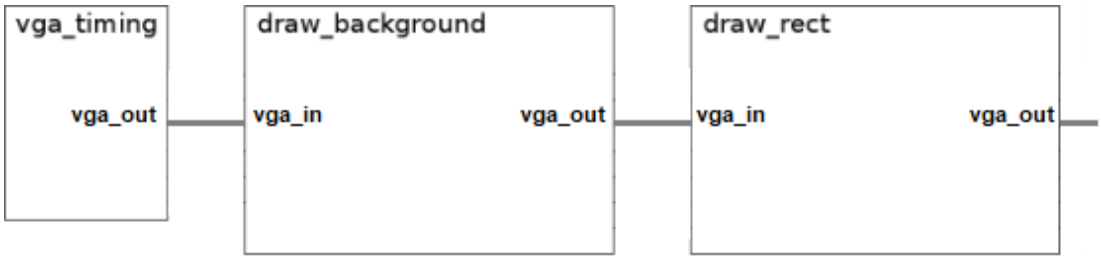
Temat ćwiczenia

Zadanie do wykonania w ramach ćwiczenia to opracowanie programu, który będzie na ekranie wyświetlał prostokąt poruszany myszą.

W poprzednim ćwiczeniu zaprojektowałeś układ rysujący na ekranie prostokąt. W tym ćwiczeniu rozbudujesz opracowany wcześniej układ tak, aby umożliwić dynamiczne sterowanie pozycją prostokąta na ekranie.

Przebieg ćwiczenia

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



W tym ćwiczeniu dodamy do schematu jeszcze dwa moduły:

- moduł sterujący położeniem rysowanego prostokąta,
- moduł obsługujący mysz.

Aby umożliwić dodanie tych dwóch modułów musimy jeszcze wcześniej wymienić na nowy generator sygnału zegara, ponieważ system wyświetlania na ekranie działa z zegarem 40 MHz, natomiast dostępny kontroler myszy działa z zegarem 100 MHz. Drugą rzecz, którą musimy zrobić, to zmodyfikować moduł rysowania prostokąta tak, aby można było mu podawać pozycję prostokąta z zewnątrz.

3. Wymiana modułu generatora zegara

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

W projekcie, w pliku **top_vga_basys3.sv** znajduje się moduł **MMCME2_BASE** (clk_in_mmcem2) wraz buforami **IBUF** (clk_ibuf), **BUFH** (clk_out_bufh), **BUFGCE** (clk_out_bufgce) oraz **ODDR** (pclk_oddr).

Wszystkie wymienione moduły z wyjątkiem ostatniego (**ODDR**, który służy do wygenerowania kopii sygnału zegara na wyjściu FPGA) zostaną zastąpione przez nowy moduł.

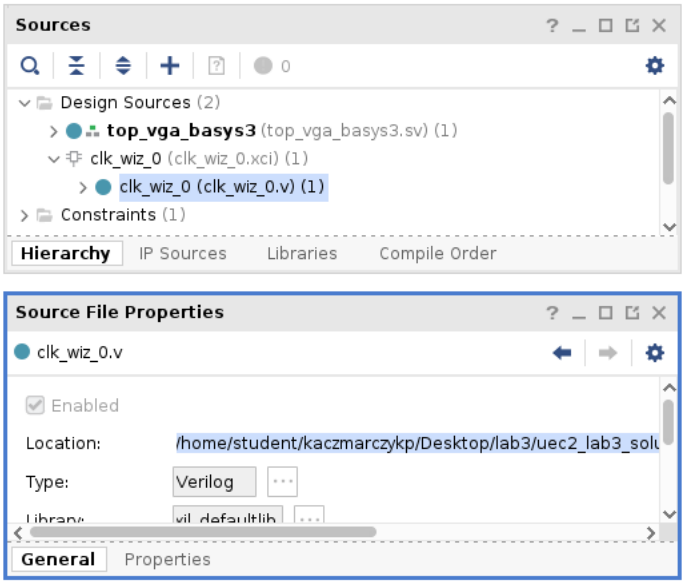
Opisy wymienionych modułów znajdziesz w dokumencie **Xilinx UG 953: 7 Series FPGA and Zynq-7000 SoC Libraries Guide** (dostępny w dokumentacji Vivado oraz na www).

*1.1. Korzystając z terminala przejdź do folderu **fpga/build** i uruchom Vivado komendą **vivado <nazwa_projektu>.xpr**. Jeśli nie posiadasz takiego folderu, uruchom najpierw skrypt generujący bitstream i poczekaj na jego zakończenie.*

*1.2. W oknie Vivado otwórz **IP Catalog** (Project Manager → IP Catalog), wyszukaj **Clocking Wizard** i uruchom go.*

- *Wykorzystaj moduł **MMCM**.*
- *Włącz opcję **Safe Clock Startup**.*
- *Nazwę sygnału wejściowego ustaw jako **clk**, a jego częstotliwość na 100 MHz.*
- *Wygeneruj dwa sygnały zegarowe o nazwach **clk100MHz** i **clk40MHz** o odpowiednich częstotliwościach.*
- *Zwróć uwagę na to, aby bufory wyjściowe **Drives** dla obydwu sygnałów były tego samego typu **BUFGCE.***
- *Odznacz wejście reset, a wyjście **locked** pozostaw zaznaczone.*
- *Po kliknięciu „OK" wybierz „Synthesis Option" **Global** i kliknij **Generate**.*

*1.3. W oknie **Sources** rozwiń zawartość wygenerowanego IP core, kliknij na znajdujący się wewnątrz plik .v, a następnie, w oknie **Source File Properties**, skopiuj ścieżkę do folderu, w którym się znajduje. Zamknij Vivado.*



1.4. Przejdź do folderu ze skopiowanej ścieżki. Przeglądnij wygenerowane pliki .v i .xdc. Te z nich, które zawierają kod (a nie tylko komentarze), skopiuj do odpowiednich folderów swojego projektu.

*1.5. W module **top_vga_basys3** usuń cztery wymienione we wstępie moduły i zastąp je jednym (odpowiednim) ze skopiowanych. Pamiętaj o poprawnym podłączeniu bufora **ODDR** oraz modułu **top_vga**. Usuń też zbędne deklaracje sygnałów (wire, logic).*

*1.6. W pliku **top_vga_basys3.xdc** zakomentuj dotychczasową definicję sygnału zegara **create_clock**. Nie będzie ona potrzebna, ponieważ wygenerowany moduł zawiera już plik XDC z tą definicją.*

1.7. Dopasuj pozostałe pliki projektu, które tego wymagają.

*1.8. Sprawdź, czy działa symulacja **top_fpga**. Jej wynik powinien być identyczny jak przed modyfikacją.*

1.9. Wygeneruj bitstream i zaprogramuj płytkę. Sprawdź, czy układ działa poprawnie.

2. Kontrola pozycji prostokąta przy pomocy myszy

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

*2.1. W module **draw_rect** zastąp parametry określające pozycję prostokąta dwoma 12-bitowymi wejściami (pamiętaj, że nazwy parametrów piszemy dużymi literami, a nazwy portów małymi).*

*2.2. Dodaj do projektu dwa pliki źródłowe dostępne na UPELu: **MouseCtl.vhd** i **Ps2Interface.vhd**. Pliki te zawierają napisany w języku VHDL kontroler obsługujący mysz w standardzie PS/2 i pochodzą z projektu demonstracyjnego dla płytki BASYS3 dostępnego na stronach www firmy Xilinx.*

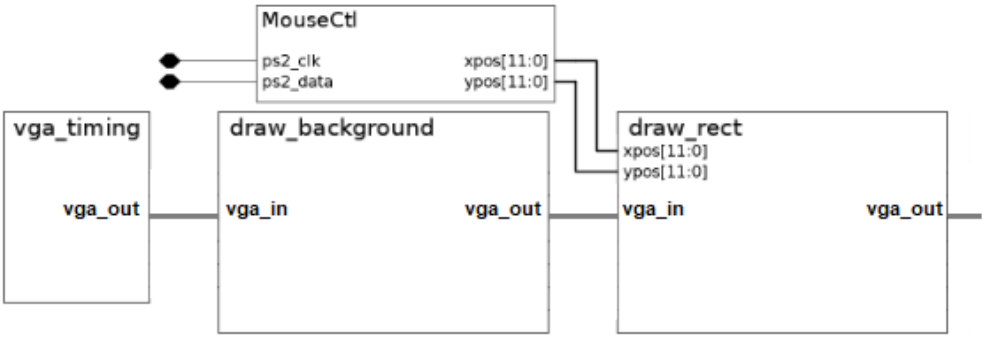
*2.3. Przeczytaj nagłówek pliku **MouseCtl.vhd**. Zawiera on opis zaimplementowanego protokołu i działania modułu.*

*2.4. Dodaj do modułu **top_vga** porty **ps2_clk** i **ps2_data**. Uwaga: kierunek portów to **inout**. Dodaj też wejście dla sygnału zegara **100 MHz**.*

*2.5. Znajdź w pliku XDC sekcję **USB HID (PS/2)** i zapoznaj się z nią. Pierwsza linijka mapuje port (PS2Clk) na konkretny pin FPGA (C17), druga ustawia standardową konfigurację **LVCMOS33**, a trzecia włącza rezystor podciągający **PULLUP**.*

W pliku XDC odkomentuj zawartość tej sekcji dla obu sygnałów. Sprawdź jakich nazw portów w niej użyto i takie same nazwy dodaj w deklaracji portów modułu **top_vga_basys3**. *Następnie zapewnij ich połączenie z modułem* **top_vga**. *Podłącz też zegar 100 MHz.*

*2.6. Dodaj do modułu **top_vga** moduł **MouseCtl** i podłącz go zgodnie ze schematem poniżej.*



*Zwróć uwagę, że moduł **MouseCtl** powinien być podłączony do zegara 100 MHz, a pozostałe do 40 MHz. Czy spodziewasz się z tego powodu jakichś problemów? (podpowiedź: Clock Domain Crossing)*

2.7. Dostosuj do wprowadzonych zmian wszystkie pliki projektu, które tego wymagają. Symulacje top_vga i top_fpga, o ile porty PS/2 pozostaną niepodpięte, powinny pokazać prostokąt w pozycji (0,0).

2.8. Skompiluj i uruchom na płytce program. Do złącza USB płytki Basys3 podłącz mysz. Na monitorze powinieneś zobaczyć zdefiniowany przez Ciebie prostokąt poruszający się zgodnie z ruchami myszy.

2.9. Przeglądnij log pod kątem ostrzeżeń (WARNING). Czy wszystkie rozumiesz?

3. Wyświetlanie kursora myszy

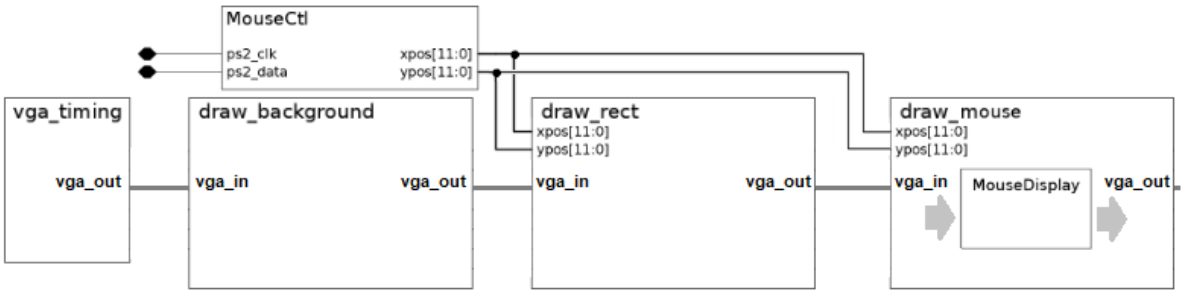
Zrób kopię zapasową projektu przed wprowadzaniem kolejnych modyfikacji, jeżeli takiej nie posiadasz.

*3. 1. Dodaj do źródeł plik **MouseDisplay.vhd** dostępny na UPELu. Jest to moduł udostępniany przez firmę Xilinx w projekcie demonstracyjnym dla płytki BASYS3, napisany w języku VHDL. Wykorzystasz go narysowania na ekranie kursora myszy.*

*3.2. Przejrzyj plik. Zwróć uwagę, że linie wejściowe RGB i blank są zakomentowane. Usuń w pliku komentarze w odpowiednich miejscach (nie tylko w definicjach portów, ale również w kodzie opisującym stan wyjść). W razie potrzeby (np. sygnalizowanej przez Warning) dodaj brakujące sygnały do sensitivity list (jest to lista sygnałów wpisanych w nawiasie po słowie kluczowym **process**; zmiana któregokolwiek z tych sygnałów uruchamia wykonanie bloku process)*

*3.3. Zwróć uwagę, że moduł **MouseDisplay** przyjmuje na wejścia prawie wszystkie sygnały z interfejsu **vga_if**, oprócz **vsync** i **hsync** (sygnał blank można uzyskać jako sumę logiczną hblnk i vblnk). Na wyjściu dostępne są jednak tylko kolory. Dodatkowo, ponieważ wyjścia red_out, green_out i blue_out są rejestrowane (przechodzą przez przerzutnik), to w celu utrzymania wyrównanych ze sobą sygnałów, konieczne jest wprowadzenia opóźnienia o jeden takt zegara na pozostałych sygnałach vga. Aby utrzymać czytelność modułu top_vga, a także, aby pisane przez nas moduły były w pełni niezależne (tzn. aby w dowolnym momencie można było dany blok vga odpiąć lub dopiąć do łańcucha, bez konieczności dodatkowych modyfikacji), całą tę funkcjonalność zamkniemy w jednym nadrzędnym module. W tym celu:*

- *Utwórz nowy moduł **draw_mouse**, podobny do draw_rect, z dokładnie takim samym zestawem portów*
- *Umieść w nim instancję **MouseDisplay** i podłącz sygnały do interfejsów (Którego z sygnałów zegara użyjesz w tym module?)*
- *Sygnały kontrolne vga opóźnij o jeden takt zegara przy pomocy przypisania w bloku **always_ff***



*3.4. Umieść w module **top_vga** instancję **draw_mouse**, zgodnie z powyższym rysunkiem.*

3.5. Dostosuj do wprowadzonych zmian wszystkie pliki projektu, które tego wymagają. Symulacje top_vga i top_fpga, powinny pokazać prostokąt w pozycji (0,0) z nadpisanym na nim kursorem.

3.6. Skompiluj i uruchom na płytce program. Powinien on wyświetlać zdefiniowany przez Ciebie prostokąt poruszający się po ekranie zgodnie z ruchami myszy, z nadpisanym kursorem.

3.7. Przeglądnij log pod kątem ostrzeżeń (WARNING). Czy wszystkie rozumiesz?

Wyniki ćwiczenia

Jako wynik ćwiczenia należy:

- zaprezentować działanie programu na następnych zajęciach laboratoryjnych i wyjaśnić słownie zasadę działania. Program powinien wyświetlać na ekranie obrazek z wcześniejszego ćwiczenia. Ponadto, pozycja prostokąta powinna być kontrolowana przez mysz, a dokładną pozycję mszy na ekranie powinien wskazywać kursor.

- załadować spakowane (ZIP) archiwum projektu na UPEL.

Archiwum powinno zawierać wszystkie pliki źródłowe i skrypty, potrzebne do wygenerowania bitstreamu i przeprowadzenia symulacji. Dodatkowo, w folderze results, powinny się znaleźć: bitstream, warning_summary oraz obrazek tiff wygenerowany przez testbench głównego modułu. Pozostałe pliki (w tym również foldery i pliki ukryte, za wyjątkiem pliku .gitignore) powinny zostać usunięte.

Projekt powinien być napisany zgodnie z zasadami opisanymi w pliku „Zasady modelowania w języku SystemVerilog pod kątem syntezy", dostępnym na UPEL. Inne style kodowania mogą być stosowane wyłącznie po podaniu źródła.

Nie załadowanie projektu w terminie podstawowym = -1 pkt do oceny
Nie załadowanie projektu w terminie ostatecznym = 0 pkt za ćwiczenie