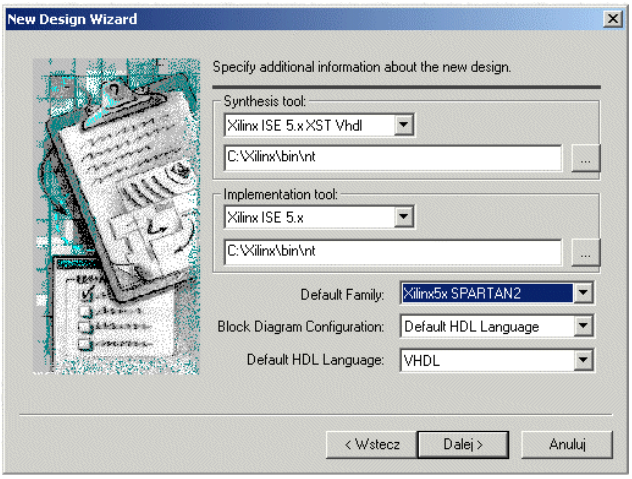
**Tutorial dla projektów dla Płyty XSA Extension**

**Autor: Ernest Jamro**

Zadaniem projektu jest zbudowanie modułu umożliwiającego wyświetlanie na wyświetlaczu 7-segmentowym wciśniętych klawiszy.

**Rozpoczęcie projektu w ActiveHDL.**
Pierwszym zadaniem jest otworzenie nowego pustego (ang. empty) projektu w ActiveHDL o następujących parametrach:




Następnym zadaniem jest skopiowanie do katalogu roboczego (katalog src) plików klawiatura.vhd, led_hex.vhd, łacznik.vhd, top.bde oraz spartsn.ucf. Pliki te znajdują się w materiałach do tego tutorialu. Alternatywnym rozwiązaniem jest zamiast skopiowanie elementu nadrzędnego top.bde skopiowanie elementu top_vhdl_vhd – w ten sposób elementem nadrzędnym nie będzie schemat a plik VHDL. Zawsze możliwe jest wygenerowanie pliku VHDL ze schematu wchodząc do menu programu BDE: Diagram/Generate HDL Code.

Następnym etapem jest dodanie powyższych plików do programu ActiveHDL poprzez wybranie menu w programie ActiveHDL Design/Add Files to design.

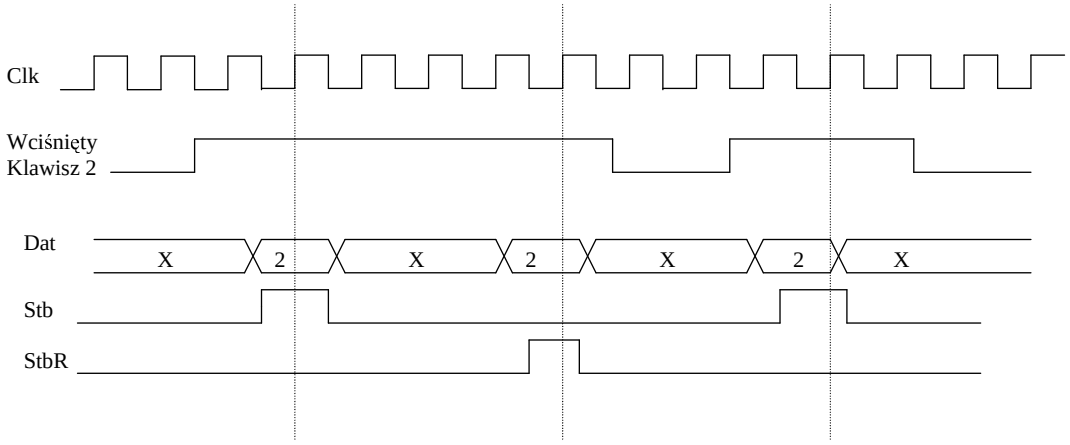
**Moduły wchodzące w skład projektu.**

**klawiatura.vhd** – plik służy do obsługi klawiatury. Plik ten należy traktować jako czarną skrzynkę – czyli należy zrozumieć znaczenie poszczególnych sygnałów ale nie należy próbować zrozumieć wewnętrznej struktury pliku.
Końcówki tego modułu które należy używać przez użytkownika:
**clk, arst** – sygnał zegarowy oraz asynchroniczny reset.
**Col, Row, ABC_in** – końcówki wyjściowe, które należy wyprowadzić na zewnątrz. Znaczenia tych końcówek nie trzeba rozumieć.
**Dat** – wyjście pokazuje, który klawisz został aktualnie wciśnięty. Wyjście to jest ważne tylko wtedy kiedy jest aktywny sygnał Stb lub StbR.
**Stb** - W przypadku naciśnięcia klawisza 0-9 lub #* generowany podawany jest kod wciśniętego klawisza na wyjściu *Dat* oraz generowany jest pojedynczy (trwający jeden cykl zegara) impuls na wyjściu *stb*. Sygnał *stb* służy jako zezwolenie zegara (Clock Enable – zobacz dodatkowy moduł łącznik). Pokazuje to poniższy rysunek na którym sygnał Dat

powinien być zatrzaskiwany wraz z narastającym sygnałem zegarowym w tylko momencie kiedy sygnał Stb jest w stanie wysokim. W języku VHDL powinno się użyć następującej składni:

```
process(clk, arst)
begin
        if arst='1' then
                        datQ <="0000";
        elsif Clk'event and Clk='1' then
                        if Stb='1' then
                                datQ <=dat;
                        end if;
        end if;
end process;
```

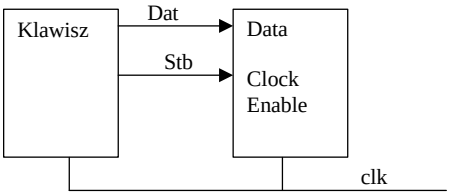
**StbR** – Podobnie jak sygnał Stb ale sygnał ten generowany jest w momencie kiedy klawisz jest przytrzymywany długo.




**Stb_ABC** - sygnał generowany podobnie jak sygnał stb ale dotyczy osobnych klawiszy A, B, C. Sygnał ten jest niezależny dla każdego klawisza.
**StbR_ABC** – podobnie jak Stb_ABC ale generowany gdy klawisz ABC jest naciśnięty długo.

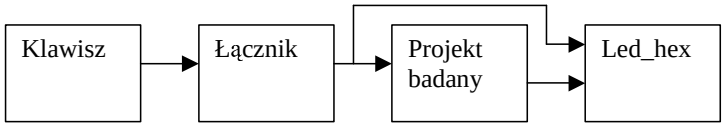
**Moduł LED_hex**
Moduł obsługujący wyświetlacz 7-segmentowy. Powinien być traktowany jako czarna skrzynka.
**clk, arst -** sygnał zegarowy i asynchronicznego resetu.
DP0, DP1, DP2, DP3 – sygnały wejściowe powodujące wyświetlenie kropli dla poszczególnych segmentów
Led0, Led1, Led2, Led3 – wejście danych do wyświetlenia dla poszczególnych segmentów. Led0 – najmniej znacząca cyfra.
Seg, Kol- wyjścia powinny być wyprowadzone na zewnątrz układu.

**Moduł lacznik.vhd**
Sygnały doprowadzone z klawiatury są sygnałami ulotnymi – czyli ich stan jest ważny tylko w momencie kiedy sygnał stb jest w stanie wysokim. Dlatego w przypadku kiedy potrzebujemy aby stan sygnału był na stale ustawiony, co jest np. potrzebne dla modułu wyświetlacza, musimy zastosować dodatkowy rejestr zatrzaskujący jego stan. Przykład takiego projektu jest podany poniżej.




W przykładowym projekcie *łącznik* zastosowano właśnie tego typu rejestry. Dodatkowo użyto rejestru przesuwnego aby umożliwić oglądanie stanu czterech ostatnio wciśniętych klawiszy. Po wciśnięciu klawisza klawisz wciśnięty jest wpisywany w prawym segmencie a wcześniej wciśnięte klawisze są przesuwane w lewo.
W normalnym projekcie moduł *łącznik* powinien być zastąpiony przez moduł użytkownika. Możliwe jest wykorzystanie tego modułu i wklejenie modułu użytkownika za modułem łącznik a pomiędzy modułem *led_hex* (wyświetlacza).

W przypadku modułów wykorzystujących nie więcej niż 8-bitów na wejściu oraz 8-bitów na wyjściu oraz wykorzystania tylko logiki kombinacyjnej (np. układ dodający) zaleca się wykorzystanie modułu łącznik oraz wyświetlanie zarówno wejść jak i wyjść na wyjść. Schemat takiego modułu może wyglądać następująco:




Następnie należy skompilować powyższe pliki poprzez wybranie menu: Design/Compile (All).

**Symulacja**
Podczas symulacji należy przetestować tylko swój projekt. Następnym etapem powinno być przetestowanie całego systemu ale ponieważ pozostałe moduły zostały już przetestowane oraz procedura testowa nie jest prosta, można zaniechać testowanie całego systemu.

**Symulacja po implementacji**
Niestety nie jest możliwe łatwe sprawdzenie poprawności działania własnego modułu po syntezie i implementacji, jeżeli jest on tylko częścią dużego projektu. Dlatego zalecane jest najpierw zsyntezowanie i implementacja tylko modułu własnego poprzez wybranie tego modułu jako modułu nadrzędnego (ang. top level). Następnie należy przycisnąć ikonę Synthesis oraz Implemntation. Należy wybrać następujący układ: Spartan2S50 TQ144 –5. Po zaimplementowaniu tylko własnego modułu można powtórnie przeprowadzić symulacje.

**Implementacja**
Następnym etapem jest synteza wszystkich elementów składowych. Dlatego należy zmienić projekt nadrzędny na top oraz powtórnie zsyntezować i zaimplementować układ. Należy pamiętać o ustawieniu podczas implementacji pliku spartan.ucf w opcjach (zakładka Translate). W pliku *.ucf znajduje się przede wszystkim przyporządkowanie poszczególnych nazw sygnałów do końcówek układu programowalnego. Należy również w opcjach

implementacji zaznaczyć Timing&Configure/ Generate Bit File aby został wygenerowany plik z rozszerzeniem *.bit który programuje układ FPGA.

**Zaprogramowanie układu FPGA**

Następnym etapem jest zaprogramowanie układu Spartan. W tym celu należy uruchomić niezależny program gxsload.exe i postępować zgodnie z opisem płyty XSA (dostępna na stronie http://galaxy.uci.agh.edu.pl/~jamro/XSA/ opis płyty XSA, str. 13-14).