Laboratorium #II-7. Projekt mikroprocesora

- *Uzupełnienie projektu prostego mikroprocesora.*
- *Programowanie zaprojektowanego mikroprocesora.*

Używane elementy: Basys3

Temat ćwiczenia.

Zadaniem do wykonania w ramach ćwiczenia jest uzupełnienie projektu prostego mikroprocesora zgodnie z podaną specyfikacją oraz napisanie programów dla tego mikroprocesora.

W wyniku wykonania ćwiczenia powinieneś otrzymać prosty mikroprocesor zaimplementowany na płytce Basys3, który po naciśnięciu przycisku mrugnie diodą wymaganą ilość razy, zgodnie z programem załadowanym przez UART. Cały program do zaświecania i gaszenia diody na określone okresy czasu ma być napisany w języku maszynowym mikroprocesora.

Przebieg ćwiczenia

Uzupełnienie kodu mikroprocesora

*Zapoznaj się z parametrami mikroprocesora przedstawionymi w załączonym pliku specyfikacji (pdf) oraz jego schematem. Twoim zadaniem będzie uzupełnienie zawartości modułu **decode** generującego sygnały kontrolne dla przetwarzania danych. Moduł ma być modułem asynchronicznym (bez sygnału zegara) i generować sygnały kontrolne **UpdateFlags**, **ALUControl** oraz **RegFileControl**.*

*Sygnał **UpdateFlags** powinien być aktywny tylko dla instrukcji arytmetycznych i logicznych, a nie aktywny dla pozostałych instrukcji. Zwróć uwagę na opis tego wejścia w module **decode** (operacje arytmetyczny aktualizują wszystkie cztery flagi, a operacje bitowe tylko flagi N i Z).*

*Sygnał **ALUControl** decyduje o funkcji wykonywanej przez przez jednostkę arytmetyczno logiczną. Zajrzyj do modułu **alu** żeby sprawdzić, jakie wartości tych sygnałów powinny być ustawione.*

*Sygnał **RegFileControl** decyduje o zapisie w rejestrach. Jest on już poprawnie zamodelowany w plikach źródłowych.*

*Załaduj do Vivado pliki źródłowe załączone do ćwiczenia. Przeglądnij schematy/kody i zastanów się, czy rozumiesz zasadę działania mikroprocesora. Moduł główny mikroprocesora to **micro**. Dostarczony jest do niego moduł symulacyjny **micro_test**, który sprawdza poprawność implementacji. Dla poprawnie napisanego kodu modułu **decode** symulator w konsoli TCL wypisze informujący o tym komunikat.*

*Przeglądnij przebiegi sygnałów. Zwróć uwagę na sygnał **extCtl** (external control). Program jest napisany w ten sposób, że czeka na sygnał **excCtl** i dopiero wtedy, gdy zarejestruje na nim stan wysoki, przystępuje do wykonania pozostałych instrukcji. Będzie to ważne w następnym punkcie, przy wykonywaniu programu krok po kroku.*

*Uwaga: W pliku **imem.v** należy ustawić poprawną ścieżkę dostępu do pliku **imem.dat**.*

*Punktacja: **2 pkt.** za poprawne wyniki symulacji i poprawny kod modułu **decode**.*

Zrób kopię zapasową projektu. :)

Uruchomienie mikroprocesora.

Celem tego kroku jest przetestowanie wykonywania programu mikroprocesora w sposób krokowy.

Aby uruchomić mikroprocesor i obserwować jego działanie będziesz potrzebował następujących modułów z poprzedniego ćwiczenia (UART): ***debounce***, ***disp_hex_mux***.

Przeglądnij moduł ***debounce****, aby zrozumieć jego działanie.*

Utwórz nowy moduł, umieść w nim mikroprocesor i pozostałe moduły.

Podłącz dwa przyciski z płytki przez moduły ***debounce*** *do wejść* ***PCenable*** *(db_tick) oraz* ***extCtl*** *(db_level) mikroprocesora. Do osobnego przycisku podłącz* ***reset***.

Jako zegar ***clk*** *ustaw odpowiedni zegar zewnętrzny 100 MHz (dodaj plik XDC).*

*Dopisz fragment kodu, który pozwoli wyświetlać na wyświetlaczu 7-segmentowym wybrane wyjścia monitorujące z procesora (****monRFData, monInstr, monPC****). Aktualnie wyświetlane wyjście ma być wybierane przez przełączniki SW.*

Wyjście ***monInstr*** *pokazuje aktualną instrukcję.*

Wyście ***monPC*** *pokazuje aktualny (8 bitów) i następny (8 bitów) stan licznika PC.*

Wyjście ***monRFData*** *wyświetla zawartość rejestru o adresie* ***monRFSrc***. *Podłącz wejście* ***monRFSrc*** *do wybranych czterech przełączników SW.*

Wymienione wyjścia pozwalają na pełny monitoring stanu procesora.

Spodziewane zawartości rejestrów dla adresów od 0 do 15 po wykonaniu programu to (heksadecymalnie):
0001, 0000, 0003, 0005, 0008, 000D, 0015, 0022, 0000, 0059, 0090, 00E9, 0179, 0262, 03DB, 0179.
(dziesiętnie: 1, 0, 3, 5, 8, 13, 21, 34, 0, 8, 144, 233, 377, 610, 987, 377 patrz kod modułu ***micro_test****).*

Punktacja: ***2 pkt.*** *za poprawnie wykonanie programu krok po kroku.*

Zrób kopię zapasową projektu. :)

Programowanie pamięci przez UART

Celem tego kroku jest uzyskanie funkcjonalności zapisywania pamięci programu przez UART.

Moduł pamięci instrukcji ***imem*** *ma zakomentowane wejścia do zapisu danych oraz funkcję zapisu. Usuń znaki komentarza i wyprowadź sygnały kontrolne na porty modułu* ***mikro*** *(dodaj nowe porty). Zakomentuj natomiast linię z instrukcją $readmemh.*

Dodaj do projektu moduł ***uart*** *z poprzedniego ćwiczenia ze wszystkim potrzebnymi modułami. Opracuj moduł kontroli zapisu pamięci, który będzie odbierał dane z* ***uart****a i wpisywał je do pamięci. Zwróć uwagę na to, że pamięć jest 16-bitowa, a dane z* ***uart****a są 8-bitowe.*

Program Tera Term pozwala na wysłanie zawartości pliku przez konkretny port szeregowy. Sprawdź poprawność opracowanego systemu przez wysłanie do pamięci zawartości pliku ***imem.bin*** *(ustaw binarny tryb transmisji). Plik ten*

zawiera zapisane binarnie komendy procesora z pliku imem.dat. Jego zawartość możesz przeglądnąć edytorem binarnym np. wxHexEditor. Po załadowaniu pamięci mikroprocesor powinien działać identycznie, jak w poprzednim punkcie.

*Punktacja: **3 pkt.** za poprawnie za wykonany program krok po kroku i poprawnie napisany moduł kontroli zapisu.*

Zrób kopię zapasową projektu. :)

Program w kodzie maszynowym.

Docelową funkcjonalnością systemu jest mrugnięcie N razy diodą po naciśnięciu przycisku. Cała funkcja mrugania ma być obsługiwana programowo.

Aby umożliwić wykonanie takiej funkcji, musisz:
*- dodać do projektu przełącznik, który umożliwi włączenie **PCenable** na stałe (zrób to tak, aby możliwe było również krokowe wykonanie programu),*
- wyprowadź na pin zewnętrzny (diodę LED) dowolny wybrany bit dowolnego rejestru.

*Napisz programy w kodzie maszynowym procesora, które mrugną diodą 2 lub 3 razy po naciśnięciu przycisku. Dioda powinna być zaświecana **dokładnie** na ½ sekundy i gaszona **dokładnie** na ½ sekundy. Zmiana liczby mrugnięć powinna być możliwa przez załadowanie pliku do pamięci przez UART, bez konieczności ponownego programowania układu FPGA. **Program ma być zapisany w postaci binarnej**.*

*Punktacja: **3 pkt**. za poprawnie działające programy (co najmniej dwa).*

Zrób kopię zapasową projektu. :)

Wyniki ćwiczenia

Jako wynik ćwiczenia należy:

*- załadować finalną wersję projekty na UPEL (katalog src i plik projektu *.xpr).*

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 pisania kodu w języku Verilog", dostępnego na UPEL (gotowych modułów nie trzeba przerabiać). Będzie to oceniane!

Wyniki będą ocenione bezpośrednio na kolejnych zajęciach.