Koder i dekoder Manchester w FPGA

Transmisja szeregowa jest bardzo chętnie stosowana wszędzie tam, gdzie niezbędne jest zmniejszenie liczby zajmowanych doprowadzeń. Najbardziej oszczędna pod tym względem jest transmisja asynchroniczna. Pojawia się tu jednak problem kodowania i dekodowania danych, gdy się nie ma do dyspozycji sygnału zegara. Rozwiązań jest kilka, między innymi oparte na układzie PLL (Phase Locked Loop) lub generatorach sygnałów zegarowych o odpowiedniej stabilności. Zajmijmy się dokładniej tym drugim przypadkiem, ponieważ nie chcemy używać układu PLL. Nawet najbardziej stabilne i precyzyjne generatory bez synchronizacji nie są w stanie na dłuższą metę umożliwić transmisji. Z tego właśnie powodu stosuje się kodowanie z zaszytym sygnałem zegarowym. Przykładem takiego kodu jest Manchester. Jest to kodowanie fazowe (Phase Encoding – PE) szeroko stosowane w aplikacjach telekomunikacyjnych.

Projekt 178

Jedną z olbrzymich zalet układów FPGA jest możliwość definiowania własnych protokołów komunikacji. Autor stanął przed problemem opracowania metody transmisji sygnału pomiarowego z przetwornika A/C po minimalnej liczbie linii światłowodowych lub LVDS na odległość kilkudziesięciu metrów. Dla zapewnienia dwukierunkowej komunikacji niezbędnym minimum jest przesłanie po jednym sygnale w każdym kierunku. Aby rozwiązanie było korzystne cenowo, przy przetworniku znalazł się niewielki układ CPLD z generatorem sygnału zegarowego oraz nadajnik. Po drugiej stronie znalazł się układ FPGA z takim samym generatorem. Ze względu na minimalną liczbę linii transmitowanie dodatkowego sygnału zegara nie wchodziło w grę. Rozwiązaniem okazała się transmisja kodowana Manchesterem.

Odbieranie danych z przetwornika

Pierwszym etapem jest odebranie danych z przetwornika, za co odpowiedzialna jest maszyna stanów przedstawiona na **list. 1**. Do wejścia START przyłączone jest sterowanie przychodzące z układu nadrzędnego. Pojawienie się stanu wysokiego na tym wejściu uruchamia sekwencję, która powoduje wykonanie jednego pomiaru. Sygnał ten nie jest generowany lokalnie, aby zapewnić synchroniczne próbkowanie w wielu modułach. Maszyna stanów generuje sekwencję sygnałów niezbędnych do odczytania danych z przetwornika i następnie wpisuje je do kodera.

Zasada działania kodera i dekodera Manchestera

Koder przystosowano do transmitowania 18-bitowego wyniku z przetwornika (**list. 2**). Wynik ten zatrzaskiwany jest stanem wysokim na wejściu *wrn*. Aktualny stan przetwornika przepisywany jest do rejestru *data*. Następnie ustawiany jest stan wysoki sygnału *go* oznaczającego pracę kodera i zerowane są liczniki odpowiedzialne za liczenie wysłanych bitów (*bitnr*) oraz preskaler (*clk_div*). W kolejnych taktach zegara generowany jest zegar transmisji *sclk*, który wynika z podziału przez 16 częstotliwości zegara, którym taktowany jest układ. Równolegle generowany jest sygnał *nrz* zawierający dane. Na wyjściu, zgodnie z zasadą kodowania, przesyłany jest sygnał, który jest wynikiem funkcji *nrz xor sclk*. Po danych zawierających wynik przetwarzania, dodawany jest bit parzystości (*parity*), generowany na podstawie sygnału *nrz*.

Zakodowanie sygnału jest stosunkowo proste, nieco trudniejsze jest jego dekodowanie, ponieważ konieczne jest odtworzenie zegara transmisji. Na **list. 3** przedstawiono dekoder komplementarny do przedstawionego wcześniej kodera.

Najważniejszą częścią dekodera jest odtworzenie sygnału zegara transmisji, co zrealizowane jest przy użyciu takiego samego preskalera jak w koderze, z tą różnica, że jest on uruchamiany po wykryciu na wejściu sygnału zakodowanego, a dokładniej dowolnego opadającego zbocza.

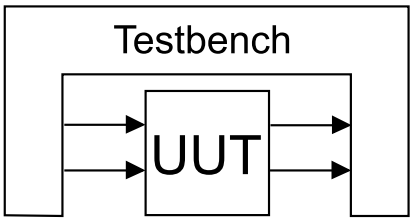
Dane z wejścia wsuwane są do rejestru złożonego *inr0*, *inr1*. Kiedy w rejestrze znajdzie się kombinacja odpowiadająca opadającemu zboczu, zostaje rozpoczęta sekwencja odbierania danych. Ustawiany jest sygnał *go* na „1", co powoduje uruchomienie preskalera oraz licznika odebranych bitów (*bitnr*). W odpowiednich chwilach czasowych, wyznaczanych przez preskaler, wartości sygnału *mdi* są zapamiętywane (dwukrotnie na okres zegara), a ich wartości służą do rekonstrukcji sygnału *nrz* oraz, na jego podstawie – bitu parzystości.

Sygnał *nrz* jest traktowany jako wejście rejestru przesuwnego taktowanego zegarem transmisji. Kiedy licznik *bitnr* osiągnie wartość b10110, dane z rejestru są zatrzaskiwane, sprawdzana jest parzystość i przesyłane są sygnały *dout, dr* i *ok*. Sygnał *go* jest zerowany.

W tym momencie koder i dekoder Manchestera są już gotowe, jednak zanim ich użyjemy, należy je gruntownie przetestować.

Kilka słów o testbenchach

Zagadnienie testowania kodu w językach opisu sprzętu jest znacznie bardziej skomplikowane, niż ma to miejsce w przypadku języka C i podobnych. Gruntowne przetestowanie modułu napisanego w VHDL-u



Rys. 1. Środowisko testowe

wymaga stworzenia modułu testującego tzw. testbencha. Nie mają żadnych wejść i wyjść, a w ich wnętrzu jest instancjonowany testowany moduł (**rys. 1**). W module takim generuje się sygnały wymuszeń oraz odbiera odpowiedzi testowanego modułu, a czasem również analizuje się ich poprawność. Jest to moduł, który nie jest poddawany syntezie, więc nie obowiązują tu ograniczenia do syntezowalnych konstrukcji języka. Pozwala to w łatwiejszy sposób generować i analizować sygnały. W przypadku testbenchy dla dużych projektów, objętość ich kodu może być większa, niż testowanego modułu. W profesjonalny pakietach oprogramowania, takich jak Acitve-HDL firmy Aldec, dostępne są narzędzia wspomagające tworzenie testbenchy oraz testowanie. Przykładem jest Code Coverage, który sprawdza, czy moduł testujący w czasie symulacji spowodował, że został wykonany każdy fragment testowanego kodu.

Jak pisać testbenche?

Odpowiedź na postawione pytanie mogłaby mieć objętość książki, dlatego zawarte tutaj informacje będą się ograniczać do absolutnych podstaw i kilku wskazówek.

Jak każdy moduł w języku VHDL, tak i testbench składa się z *entity* i *architecture*, z tą różnicą, że to pierwsze jest puste. Nie występują tu deklaracje portów, bo moduł nie komunikuje się z otoczeniem (rys. 1).

W testbenchu należy wygenerować sygnały niezbędne do pracy testowanego modułu. Zwykle takimi sygnałami są zegar i reset. Przykłady generowania tych sygnałów przedstawiono na **list. 4**.

Do generowania sygnałów można wykorzystać opóźnienia asynchroniczne, których używanie w syntezowanych modułach mija się z celem, ponieważ nie mogą zostać zsyntezowane, więc są ignorowane przez narzędzia do syntezy. W testbenchu można również bezpiecznie używać zmiennych typu *variable*, ponieważ nie wprowadzą tu różnic pomiędzy wynikami symulacji funkcjonalnej a działaniem kodu po syntezie.

Na koniec można sprawdzić automatycznie, czy moduł odpowiada poprawnie i wyświetlić wynik w konsoli. Tutaj należy się uwaga: wypisywanie dużych ilości tekstu w konsoli może znacząco obniżyć szybkość symulacji. W przypadku małych projektów może nie robić to różnicy, jednak dla dużych projektów będzie to zasadniczym problemem. Z tego powodu narzędzia do symulacji zostały wyposażone w możliwość blokowania komunikatów wyświetlanych w konsoli. Blokowanie odbywa się na poziomie ważności komunikatu (*NOTE*, *WARNING*, *ERROR*, *FAILURE*). W wyniku blokady komunikatów np. typu *note* możemy ich nie zobaczyć, mimo iż zostały wygenerowane.

Testbench dla modułu kodera/ dekodera Manchester

Na **list. 5** przedstawiono moduł testbencha. Jak widać, nie ma żadnych portów, więc sekcja *entity* jest pusta. W module instancjonowany jest komponent *top*, który przedstawiono na **list. 6**. Jego zadaniem jest połączenie modułu obsługi przetwornika oraz kodera i dekodera Manchestera. Opiszemy go w VHDL strukturalnym.

W testbenchu, oprócz przedstawionych wcześniej sekcji generującej sygnały zegara i zerowania, znalazła się sekcja, która symuluje przetwornik analogowo-cyfrowy. W procesie *St* generowane są sygnały Start – symulujący komendę rozpoczęcia przetwarzania, DA – interpretowany jako dane z przetwornika oraz Busy – który przetwornik wystawia w czasie konwersji. W procesie tym użyto również polecenia *assert* do sprawdzenia czy dane na wejściu i wyjściu są takie same oraz czy sygnał *ok* jest poprawnie ustawiany. Należy tutaj pamiętać o warunku, po spełnieniu którego przesyłane są dane: *DA=DOUT and ok='1'*. Zapis oznacza, że w konsoli pojawi się informacja, jeżeli warunek nie zostanie spełniony. Jeżeli transmisja będzie poprawna, to podczas symulacji nie pojawi się żadna informacja. Błąd sygnalizowany będzie komunikatem zbliżonym do przedstawionego na **rys. 2**.

Taki sposób informowania o wyniku końcowym symulacji pozwala w łatwy sposób badać, z jak dużą różnicą częstotliwości dwóch generatorów sygnału zegarowego może pracować układ. W testbenchu znalazły się dwie niezależne sekcje generujące zegar dla kodera i dekodera, co pozwala zasymulować rozbieżność częstotliwości generatorów. Z prób przeprowadzonych przez autora wynika, że granicą jest ok. 0,5%. Przy większej różnicy pojawiają się błędy transmisji. Do przesyłania słowa o długości 18 bitów potrzebne jest ponad 350 okresów zegara. Przy różnicy częstotliwości na poziomie 0,5% szybszy z zegarów ma o prawie 2 okresy więcej. Zmniejszenie długości słowa na pewno podniesie odporność na różnice częstotliwości sygnałów zegarowych.

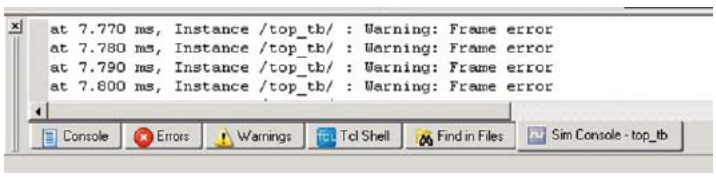
Wyniki działania testbencha przedstawiono na **rys. 3**. Między niebieskimi kreskami znajduje się jedna pełna ramka danych. Przedstawiony testbench pozwala wygenerować dowolnie długą symulację, np. taką, w której będzie 100 tys. ramek na sekundę symulacji. Mamy tu możliwość obejrzenia zakodowanego przez nas sygnału (*coded*). Moduły kodera i dekodera połączone są tylko jedną linią danych (**list. 6**) i pracują z różnymi sygnałami zegarowymi.

Podsumowanie

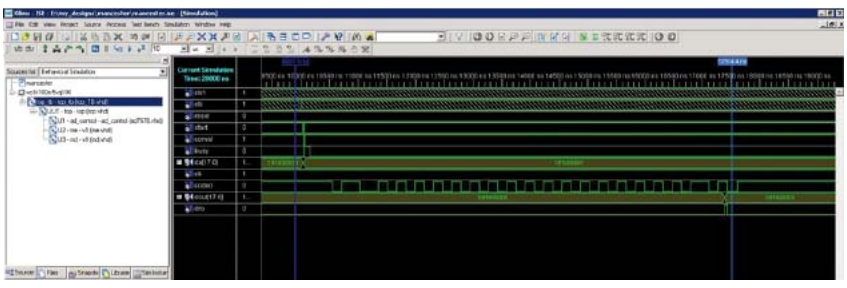
W ten sposób powstało coś na kształt IP Core umożliwiającego transmisję szeregową pomiędzy dwoma układami FPGA. Na kształt, bo profesjonalnie przygotowane i sprzedawane *IP Core* muszą być bardzo szczegółowo przetestowane oraz wyposażone w odpowiedni zestaw skryptów do symulacji. Na płytce dołączonej do wydania umieszczono projekt przygotowany w ISE WebPack, zawierający symulator, na którym można przeprowadzić własne próby.

Autor stosuje opisane moduły do przesłania danych z przetwornika analogowo-cyfrowego do układu akwizycji danych. Nic nie stoi na przeszkodzie, aby wykorzystać moduł kodera i dekodera, i zbudować tor transmisji dwukierunkowej w dowolnym systemie opartym o układy FPGA lub CPLD.

Artur Boroń
artur.boron@agh.edu.pl



Rys. 2. Ostrzeżenia w konsoli symulatora z pakietu ISE WebPack



Rys. 3. Wyniki działania testbench-a