Adam MILIK
POLITECHNIKA ŚLĄSKA, INSTYTUT ELEKTRONIKI
ALDEC ADT

Modelowanie i weryfikacja złożonych układów cyfrowych z wykorzystaniem środowiska MATLAB i Simulink

Dr inż. Adam MILIK

Ukończył studia na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej. Pracę doktorską obronił w 2003 r. Jest adiunktem w Instytucie Elektroniki Politechniki Śląskiej. Jego zainteresowania naukowe to układy logiki programowalnej, sterowniki programowalne, modelowanie i synteza złożonych układów sprzętowo-programowych.

e-mail: adam.milik@polsl.pl

Streszczenie

Artykuł przedstawia metodę połączenia środowiska modelowania matematycznego MA-TLAB oraz Simulink ze środowiskiem przeznaczonym do modelowania systemów cyfrowych opartym na symulatorze zgodnym z normami języków Verilog i VHDL. Zaproponowane rozwiązania umożliwiają łatwą weryfikację i projektowanie urządzeń cyfrowych z wykorzystaniem podejścia algorytmicznego.

Słowa kluczowe: symulacja, symulacja wspólna, modelowanie, MATLAB, Simulink.

Complex Digital Circuits Verification with use of MATLAB and Simulink

Abstract

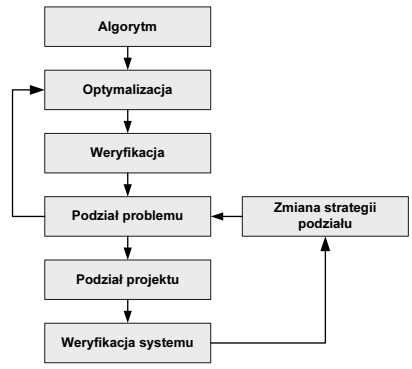
The paper presents methodologies of interfacing a standard HDL simulation environment to MATLAB and Simulink environments. Two co-simulation environments are considered for MATLAB and Simulink respectively. There are several factors that must be resolved before smooth and efficient cosimulation is possible. In case of complex circuits performance is extremely important factor. There are considered problems of time synchronization, data conversion and communication between two different simulation environments. As simulator kernels are inaccessible for modification the optimization can only be introduced to linking library.

Keywords: Simulation, Co-simulation, Modeling, MATLAB, Simulink.

1. Wprowadzenie

Rosnąca złożoność współczesnych układów i systemów cyfrowych wymaga odpowiednich narzędzi do weryfikacji ich działania. Złożone algorytmy przetwarzania informacji czy też sterowania są projektowane z wykorzystaniem podejścia systemowego (rys. 1). Projekt rozpoczyna się od sformułowania algorytmu i jego weryfikacji. Faza to jest prowadzona z wykorzystaniem narzędzi odpowiednich do wybranego poziomu abstrakcji. Uzyskanie zadawalającego algorytmu pozwala na przejście do dalszego etapu implementacji. Podczas wstępnej fazy projektowania algorytmu zostaje opracowany zbiór układów testowych umożliwiających wszechstronną weryfikację opracowanego algorytmu [1, 2, 3].

Proces automatycznej syntezy układów cyfrowych w oparciu o języki opisu sprzętu nakłada ograniczenia na poziom abstrakcji opisu. Abstrakcyjny opis musi zostać ograniczony do poziomu przesłań międzyrejestrowych (ang. RTL). W konsekwencji następuje znaczna dysproporcja pomiędzy modelem opisanym na poziomie algorytmicznym a modelem układu podlegającym syntezie. Proces weryfikacji wymaga przekształcenia generatora oraz analizatora odpowiedzi tak, aby mógł zostać zapisany za pomocą konstrukcji języka opisu sprzętu i funkcjonować w testowanym układzie. Przedstawione podejście prowadzi praktycznie do całkowitego rozdziału pomiędzy procesem projektowania algorytmicznego a procesem projektowania układu sprzętowego [2, 3].



Rys. 1. Metodologia projektowania układów na poziomie systemu
Fig. 1. System level design methodology

W niniejszym artykule zostanie omówiona metodologia oraz przedstawione rozwiązania pozwalające na połączenie środowisk modelowania algorytmicznego ze środowiskiem modelowania układów cyfrowych opisanych językami opisu sprzętu dalej zwanymi językami HDL.

2. Środowisko modelowania algorytmicznego

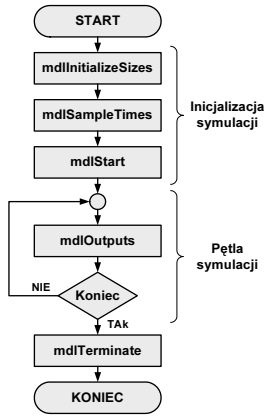
Złożoność opracowywanych algorytmów jest na tyle wysoka, że wymaga odpowiedniego wsparcia w czasie implementacji i weryfikacji. Jednym z najbardziej rozpowszechnionych w gronie naukowym i inżynierskim środowisk modelowania algorytmicznego jest pakiet MATLAB wraz z pakietem Simulink [4]. Obecny od ponad dwóch dekad stał się rozwiązaniem standardowym. Środowisko modelowania matematycznego jest odpowiednie do rozwiązywania różnych problemów technicznych od przetwarzania sygnałów do sterowania procesami. Analiza może być prowadzona w dziedzinach układów ciągłych i dyskretnych w czasie. Obok zdolności modelowania środowisko jest wyposażone w pełny zestaw narzędzi do analizy i wizualizacji projektu. Istotną cecha środowiska MATLAB jest możliwości definiowania własnych funkcji napisanych za pomocą wbudowanego języka M (M-Function) oraz języków wysokiego poziomu C i Fortran (MEX-Function).

Pakiet Simulink aktualnie stanowi nieodłączny element środowiska MATLAB. Przeznaczony jest do modelowania i analizy systemów dynamicznych. Pozwala na modelowanie całego spektrum systemów o przetwarzaniu liniowym i nieliniowym jak i sposobie próbkowania ciągłym, dyskretnym lub mieszanym. Projektowanie układów odbywa się z wykorzystaniem bloków funkcjonalnych. Podobnie jak MATLAB pozwala na definiowanie bloczków o funkcjonalności zdefiniowanej przez użytkownika. Stwarza to możliwość włączenia do systemu symulacji nowych elementów funkcjonalnych. Rozszerzenie możliwości środowiska o bloki, których opis jest reprezentowany za pomocą języków HDL pozwoli na spójne prowadzenie projektu i weryfikację realizacji zaprojektowanego algorytmu bezpośrednio w środowisku algorytmicznym. Pozwala to uniknąć wielu niedogodności oraz błędów związanych z przejściem pomiędzy środowiskami symulacyjnymi.

3. Reprezentacja funkcjonalna modułów sprzętowych w środowisku Simulink

Simulink pozwala na utworzenie bloczka, którego funkcjonalność jest opisana za pomocą funkcji S [4, 7]. Funkcja S rejestruje w symulatorze zestaw funkcji. Będą one wywoływane przez jadro symulatora Simulink w różnych momentach procesu symulacji

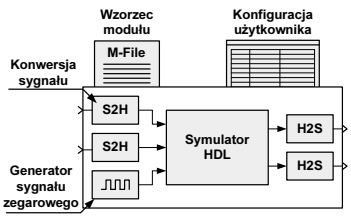
w celu zapewnienia poprawnego przebiegu procesu modelowania całego systemu dynamicznego. Schemat blokowy przedstawiający kolejność wywołania poszczególnych funkcji został przedstawiony na rysunku (rys. 2).



Rys. 2. Wywoływane podfunkcje w czasie procesu symulacji w symulatorze Simulink
Fig. 2. Function call sequence in Simulink simulation process

Wprowadzenie modułu sprzętowego symulowanego za pomocą symulatora HDL do środowiska Simulink wymaga zaimplementowania zestawu cech funkcjonalnych umożliwiających jego właściwe osadzenie na schemacie modelu oraz prowadzenie procesu symulacji. Aktualnie dostępne są rozwiązania dla symulatorów MTI oraz Cadence [7]. Przedstawione rozwiązanie zostało opracowane dla symulatora rodziny ActiveHDL firmy Aldec [8]. Wyróżnia je z pośród dostępnych rozwiązań ogromna prostota użycia i znacznie lepsza wydajność procesu kosymulacji. Interfejs umożliwia pełną kontrolę nad symulowanym obiektem co pozwala na dostosowanie do różnych potrzeb symulacji. Uzyskanie powyższego rozwiązania wymagało zaproponowania schematu procesu kosymulacji oraz zaprojektowania sprawnego mechanizmu wymiany informacji pomiędzy symulatorami.

Moduł sprzętowy udostępniany do symulacji w środowisku Simulink musi zostać przekazany wraz z podstawowymi informacjami o interfejsie, parametrach, bibliotece, niezbędnych deklaracjach. Informacja taka przekazywana jest w postaci M-funkcji zwracającej strukturę opisującą moduł sprzętowy. Opis taki stanowi wzorzec dla wszystkich bloczków potomnych. Następnie za pomocą dialogu parametryzującego, projektant ma wpływ na zachowanie bloku w przestrzeni modelu. Pozwala to definiować sposób konwersji sygnałów wejściowych, typ sygnałów wyjściowych, wewnętrzne sygnały zegarowe oraz synchronizujące i inne sygnały sterujące. Dodatkowo można zadeklarować zapis przebiegów poszczególnych sygnałów czy też zatrzymanie symulatora i przejście do trybu diagnostycznego na wskutek zmian sygnału.



Rys. 3. Osadzenie i konfiguracja modułu w modelu Simulink
Fig. 3. Module placement and configuration in Simulink

W celu koordynacji całego procesu symulacyjnego zaprojektowano specjalny bloczek, którego zadaniem jest przenoszenie dedykowanych ustawień procesu symulacji wspólnej. W szczególności odpowiada on za ustawienie relacji czasowych pomiędzy oboma symulatorami, sposób prowadzenia procesu symulacji, wymuszanie trybu diagnostycznego czy też rejestrowanie pobudzeń i odpowiedzi w celu dokonania późniejszej symulacji weryfikującej bez obecności środowiska Simulink.

Przedstawione opcje maja na celu ułatwienie prowadzenia procesu symulacji i diagnostyki pozwalając projektantowi na znaczną swobodę i łatwość dostosowania do swoich potrzeb.

3.1. Przygotowanie procesu symulacji

Proces symulacji w środowisku Simulink został przedstawiony na rysunku (rys. 2). Każdy bloczek sprzętowy oraz koordynator procesu symulacji wykorzystuje identyczną S-funkcje zapewniającą jego poprawne modelowanie. Należy jednak zwrócić uwagę, że funkcja może opisywać praktycznie dowolne moduły sprzętowe o znacznie zróżnicowanym interfejsie. W przypadku ogólnym należy również założyć występowanie kilku bloczków sprzętowych na jednym modelu w programie Simulink. Podczas fazy inicjalizacji bloczka zostaje wywołanych kilka podfunkcji. Faza inicjalizująca symulację w ograniczonym stopniu jest wykonywana podczas aktualizacji modelu. Nakłada to pewne ograniczenia na przygotowanie procesu symulacji wspólnej. Podczas wstępnej rejestracji bloczka w funkcji mdlInitializeSizes dokonuje się rejestracji liczby wejść oraz wyjść. W celu zapewnienia pełnej swobody łączenia bloczka z dowolnymi sygnałami, wejście dziedziczy typ po sygnale wejściowym. W przypadku wyjść rejestracji podlega typ zadeklarowany przez użytkownika. Podczas wywołania funkcji mdlSapmleTimes zostaje zarejestrowany sposób próbkowania oraz określony kwant czasu, z jakim bloczek będzie wywoływany.

Procedurę inicjalizacyjną kończy wywołanie funkcji mdlStart. W funkcji tej każdy bloczek rejestruje własne ustawienia niezbędne do wygenerowania modułu najwyższego poziomu dla symulatora języków opisu sprzętu. Po rejestracji ostatniego z bloczków przystępuje się do utworzenia modułu najwyższego poziomu. Po czym rozpoczyna się proces tworzenia projektu w symulatorze sprzętu. Po poprawnej kompilacji projektu zostaje zainicjalizowana symulacja w symulatorze sprzętowym.

3.2. Proces symulacji wspólnej

Aby mogła rozpocząć się symulacja wspólna należy umożliwić wymianę informacji pomiędzy symulatorami. Wymaga to użycia komunikacji między procesami. Z racji, że Simulink jest dominującym symulatorem on będzie odpowiedzialny za generowanie zleceń. Zadaniem symulatora HDL jest przetwarzanie otrzymanych zleceń. Żaden z symulatorów języków opisu sprzętu nie implementuje interfejsu do efektywnego sterowania procesem symulacji za pomocom znanych metod komunikacji międzyprocesowej. W celu rozwiązania problemu komunikacji zostały napisane dwa moduły przesyłania informacji między procesami. Pomijając szczegóły nawiązania połączenia, należy zapewnić możliwość wymiany znacznej liczby niewielkich pakietów informacyjnych. Przeprowadzone badania wydajności wykazały, że systemy wielodostępne mogą przesłać przez połączenie międzyprocesowe praktycznie stałą liczbę niewielkich pakietów danych (nieznacznie zależną od częstotliwości pracy mikroprocesora). Powyższy test modelowy odzwierciedla typową sytuację wymiany informacji w procesie symulacji pomiędzy symulowanymi obiektami o wzajemnych zależnościach. W celu zapewnienia wydajnej symulacji wspólnej należy dążyć do ograniczenia liczby przesłań informacji pomiędzy symulatorami. Jak wykazało doświadczenie, głównym ograniczeniem jest liczba pakietów wysyłana w jednostce czasu rozmiar pakietu ma znacznie mniejszy wpływ. Należy zatem opracować metodę synchronizacji procesów symulacji przy możliwie najmniejszej liczbie cykli wymiany. W procesie symulacji wielu instancji modułów sprzętowych można wyróżnić trzy zasadnicze transakcje realizowane pomiędzy symulatorami. Wykonuje się je podczas wywołania funkcji mdlOutputs odpowiedzialnej za wyznaczenie stanu wyjść bloku w bieżącej chwili czasu symulacyjnego. Operację rozpoczyna się od synchronizacji czasu symulacyjnego pomiędzy symulatorami. Jeżeli wykryto rozbieżność czasu należy uruchomić proces symulacji sprzętu w celu synchronizacji domen czasowych. Następnym krokiem jest przesłanie stanu każdego z wejść. W końcowym etapie zostają pobrane

stany wyjść. Wykonywanie przedstawionej sekwencji zdarzeń na zasadzie zlecenie-odpowiedź będzie wymagało wielu przesłań międzyprocesowych, co spowoduje znaczące ograniczenie wydajności obliczeniowej. Z punktu widzenia symulatora Simulink wywołanie funkcji odpowiedzialnej za aktualizację wyjść nie narzuca kolejności wymiany informacji pomiędzy symulatorami. Grupując zlecenia w kolejkę można utworzyć pakiet zleceń do przetworzenia przez symulator. Kolejka zleceń zostaje przesłana do symulatora jako pojedyncza transakcja. Symulator odpowiada na wysłane zlecenia również w postaci kolejki gdzie wpisuje oczekiwane wynik realizowanych poleceń oraz ewentualne informacje o błędach. W przypadku bloku nie posiadającego wyjść nie jest konieczne przesłanie pakietu do symulatora. Aby zapewnić zbieżność procesów symulacyjnych zawartość kolejki jest wysyłana w przypadku, gdy konieczna jest aktualizacja wyjścia, kolejka została całkowicie zapełniona lub symulacja dobiegła końca a w kolejce znajdują się nieprzetworzone zlecenia (wywołano funkcje mdlTerminate bez sygnalizacji błędu). Przedstawione podejście pozwala na daleko idące ograniczenie liczby przesłań międzyprocesowych, co pozwala na naprzemienną ciągłą pracę jednego symulatorów z bardzo niewielkim czasem oczekiwania na przesłanie informacji.

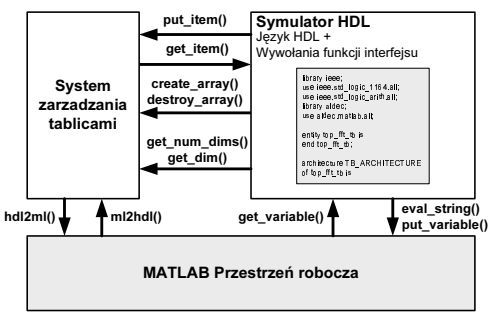
Istotnym aspektem jest również wbudowanie efektywnych procedur konwersji formatu danych pomiędzy symulatorami. Znaczna liczby obsługiwanych typów liczbowych wymaga zdefiniowania funkcji konwersji dla każdego z nich. W celu umożliwienia natychmiastowego wyboru funkcji konwersji w procesie rejestracji zostaje związana odpowiednia funkcja konwersji z każdym z wejść i wyjść bloczka.

4. Dołączenie środowiska MATLAB do symulatora układów cyfrowych

Języki opisu sprzętu pozbawione są wyrafinowanych funkcji matematycznych czy wizualizacji, które są dostępne w środowisku MATLAB. Umożliwienie realizacji obliczeń z poziomu symulatora sprzętowego bezpośrednio w środowisku MATLAB znacząco podniosłoby jego cechy funkcjonalne ułatwiając prowadzenie modelowania i analizę złożonych układów czy też pozwalając na stworzenie układów testowych wspomaganych specjalizowanym aparatem matematycznym.

Środowisko MATLAB udostępnia interfejs wywoływania zdalnego zleceń. Interfejs ten pozwala na pracę ze środowiskiem MATLAB traktując go jako serwer obliczeniowy. W przypadku przyłączenia do symulatora HDL należy rozwiązać kwestie przesyłania zmiennych oraz kierowania zleceń obliczeniowych.

Rozszerzenie własności funkcjonalnych środowisk HDL musi odbyć się bez ingerencji w jądro symulatora czy kompilatora. Najwcześniej możliwość dołączenia dodatkowych funkcji stworzył język Verilog poprzez definicję interfejsu PLI [5]. Podobny interfejs został zdefiniowany dla języka VHDL – VHPI [6]. Korzystając z tych interfejsów można wzbogacić symulator o dodatkowe funkcje bez konieczności ingerencji w jego wewnętrzne struktury. Schematycznie działanie interfejsu przedstawiono na rysunku (rys. 4).



Rys. 4. Schematyczny zestawienie funkcji interfejsu do środowiska MATLAB
Fig. 4. The MATLAB interface functions set

Pierwszym podstawowym elementem jest możliwość kierowania poleceń do środowiska MATLAB (eval_string). Polecenia są kierowane w postaci łańcuchów znakowych. Jednakże dopiero przesyłanie zmiennych pomiędzy środowiskami pozwala na wykorzystanie całego bogactwa obu środowisk i stanowi kluczowy element rozwiązania.

4.1. Przesyłanie zmiennych pomiędzy środowiskami

Przesyłanie zmiennych pomiędzy środowiskami musi odbywać się w dwóch kierunkach (get_variable, put_variable). Dotyczy to jedynie zmiennych skalarnych. Niestety w swojej koncepcji podstawowej MATLAB został pomyślany jako program operujący na danych tablicowych. Z punktu widzenia symulatorów HDL wykorzystanie takiego formatu danych jest niewygodne gdyż wymaga znacznych ilości pamięci. Należy zatem rozwiązać problem operowania zmiennymi tablicowymi poza symulatorem HDL. Został on rozwiązany poprzez użycia lokalnego repozytorium, w którym przechowywane są zmienne tablicowe. Z poziomu języka HDL można operować na elementach tablicy. Przechowywane tablice posiadają unikalne identyfikatory, które są zwracane podczas pobrania lub utworzenia tablicy. Z kolei ze środowiskiem MATLAB jest wymieniana cała tablica (hdl2ml, ml2hdl). Należy podkreślić, że zachodzi tutaj międzyprocesowa wymiana informacji. Poprzez przesyłanie całych tablic można znacząco ograniczyć czas konieczny na wymianę danych. Pozwala to na podniesienie wydajności symulacyjno obliczeniowej środowiska.

Interfejs został wyposażony w bogaty zbiór funkcji pozwalających na konwersję informacji pomiędzy typami numerycznymi a typami używanymi w symulatorze HDL. Problem ten podobnie jak w przypadku środowiska Simulink został rozwiązany przez stworzenie efektywnych funkcji konwersji zapewniających szybkie przeliczania wartości pomiędzy odpowiednimi reprezentacjami.

5. Podsumowanie

Przedstawione rozwiązania zostały zaimplementowane w pakiecie kosymulacyjnym firmy Aldec i stanowią część pakietu Active-HDL oraz Riviera.

Prace projektowe i optymalizacyjne prowadzone przy konstruowaniu interfejsu miały na celu uzyskanie możliwie najlepszej wydajności obliczeniowej oraz łatwości użycia. Interfejs do programu Simulink wykazuje około 50 – 60% większą wydajność obliczeniową w stosunku do rozwiązania dla symulatora MTI. Interfejs do programu MATLAB dla symulatora HDL dzięki zdolności operowania na elementach tablicowych pozwala na znaczną optymalizację wymiany danych. W przypadku posługiwania się tablicą o 4mln elementów operacje skalarne zabierają około 600 razy więcej czasu aniżeli jednokrotne przesłanie tablicy.

6. Literatura

[1] Ferrari A., A. Sangiovanni-Vincentelli, System Design. "Traditional Concepts and New Paradigms". Proceedings of the 1999 Int. Conf. On Comp. Des, Oct 1999, Austin
[2] Gajski, D., SpecC: Specification Language and Methodology. KAP, Norwell MA, 2000
[3] Milik A. "System level design in System-C using transaction-level modelling", Embed-ded Systems, March 2005.
[4] Hanselman D.C., B. Littlefield "Mastering MATLAB 7", Prentice Hall, New York 2005
[5] IEEE, "IEEE Standard Hardware Description Language Based on the Verilog® Hard-ware Description Language", IEEE 1995
[6] Ashenden P.J. "The Designer's Guide to VHDL, 2nd Edition", MK Publishers, San Francisco 2001
[7] Simulink: http://www.mathworks.com/products/simulink/
[8] Active-HDL: http://www.aldec.com/products/active-hdl/

Artykuł recenzowany