Wstęp

Początki języka Verilog sięgają wczesnych lat 80. ubiegłego stulecia. Głównym twórcą pierwotnej wersji języka był Phil Moorby – pracownik amerykańskiej firmy Gateway Design Automotion, zajmującej się wytwarzaniem oprogramowania CAE. Firma ta w roku 1984 rozpoczęła sprzedaż symulatora układów cyfrowych pod nazwą Verilog (nieco później Verilog XL). W tamtym okresie nie istniał jeszcze żaden standard języka opisu sprzętu (VHDL również dopiero się rozwijał i nie stanowił jeszcze standardu IEEE), dlatego też Gateway stworzył własny język i nazwał go Verilog HDL. Język Verilog przeznaczony był wówczas wyłączenie do symulacji układów cyfrowych.

W kolejnych latach symulator Verilog XL stawał się coraz popularniejszy w środowisku inżynierów. W roku 1987 firma Synopsys wykorzystała język Verilog jako język specyfikacji projektu (język wejściowy) dla narzędzi syntezy. W ten sposób Verilog zyskał zastosowanie już nie tylko jako język przeznaczony do celów symulacji, ale również do celów syntezy logicznej. Jednocześnie w tym samym roku język VHDL uzyskał standard IEEE (Instytutu Inżynierów Elektryków i Elektroników). Wraz z pojawieniem się standardu VHDL zwrócono uwagę na możliwości projektowania i syntezy układów cyfrowych metodą *top-down* (w skrócie: od ogółu do szczegółu – projektant specyfikuje projektowany system na wysokim poziomie abstrakcji, a odpowiednie narzędzia syntezy przekształcają ten opis do najniższego poziomu bramek logicznych) z zastosowaniem konstrukcji behawioralnych języków opisu sprzętu i odpowiednich narzędzi syntezy. Wspomniane wyżej czynniki spowodowały kolejny wzrost akceptacji symulatora Verilog XL wśród projektantów.

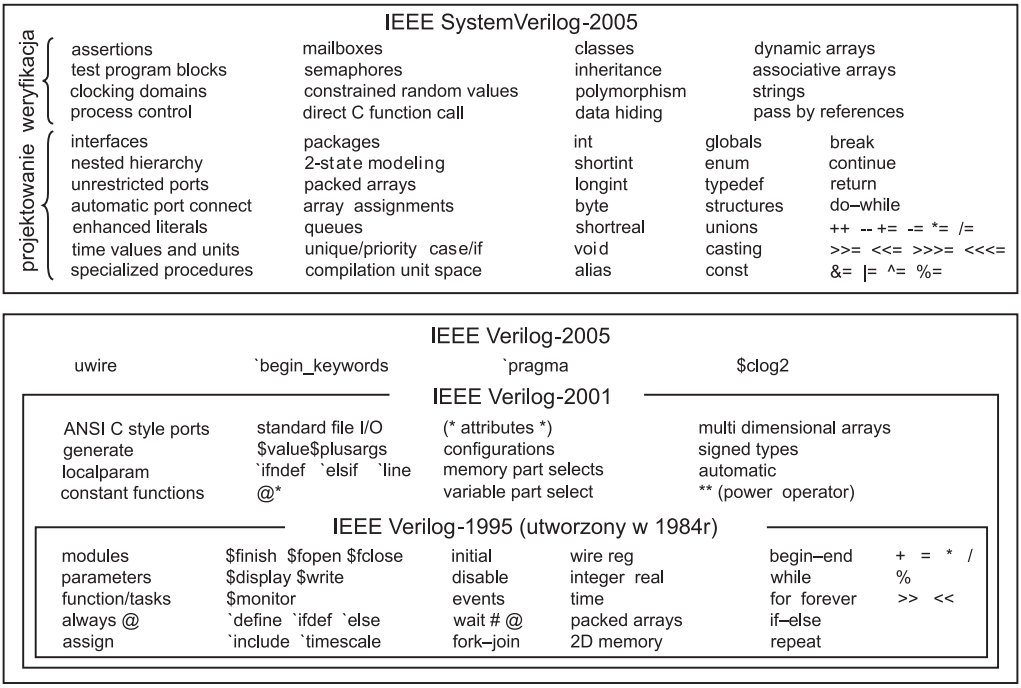
W roku 1989 firmę Gateway kupił producent oprogramowania EDA – firma Cadence. Nowy właściciel oddzielił sam język opisu sprzętu Verilog od symulatora Verilog XL, czyniąc z nich niezależne produkty. W tym samym czasie kilka innych firm kupiło prawa do używania języka Verilog, który był do tej pory językiem prawnie zastrzeżonym.

W roku 1990 firma Cadence udostępniła język Verilog publicznie, zezwalając na jego stosowanie przez inne firmy bez specjalnych licencji. Takie posunięcie było częściowo spowodowane chęcią konkurowania z językiem VHDL, który nie był produktem prawnie zastrzeżonym. W zasadniczej mierze zostało to jednak niejako wymuszone przez środowisko użytkowników Veriloga, które odczuwało potrzebę wymiany między sobą opracowanych modeli oraz wiedzy o samym języku, co przy jego prawnym zastrzeżeniu nie było rzeczą prostą.

Jednocześnie również w roku 1990 sformowano specjalną grupę pod nazwą OVI (Open Verilog International), składającą się z użytkowników Veriloga i producentów oprogramowania CAE, której zadaniem było kontrolowanie specyfikacji, i promowanie języka Verilog. W roku 1993 grupa OVI ogłosiła nową specyfikację języka, zawierającą kilka jego ulepszeń. Jednocześnie do IEEE wysłano zgłoszenie rejestracji standardu języka Verilog. Dwa lata później język Verilog uzyskał standard IEEE 1364-1995.

Kolejną nowelizację języka, zawierającą wiele istotnych ulepszeń IEEE ogłosiła w roku 2001 (IEEE 1364-2001). Jednak już rok wcześniej rozpoczęto prace pod auspicjami grupy Accelera (konsorcjum firm zajmujących się projektowaniem systemów cyfrowych oraz dostawców oprogramowania EDA) nad następną wersją języka Verilog. Efektem pracy grupy Accelera stał się System Verilog 3.0. Produkt ten nie był jednak kompletnym standardem. Stanowił jedynie zbiór rozszerzeń standardu IEEE 1364-2001. Oznaczało to, że projektanci stosujący System Veriloga musieli korzystać z dwóch podręczników użytkownika: jednego do podstawowego standardu języka i drugiego dla rozszerzenia System Verilog. Pierwotnym założeniem i jednocześnie celem, do którego dążyła grupa Accelera było połączenie tych dwóch podręczników do postaci jednego dokumentu.

W roku 2004 Accelera dodała kilka rozszerzeń do dotychczasowej wersji System Verilog i jednocześnie przekazała tę nową wersję do standaryzacji przez IEEE. Instytut IEEE postanowił jednak odłożyć włączenie System Verilog do standardu języka Verilog. Zamiast tego w roku 2005 ukazała się para standardów bazujących na języku Verilog: Verilog-2005 (IEEE 1364-2005) oraz System Verilog-2005 (IEEE 1800-2005). Niemniej jednak tuż po tym fakcie dyskusje nad połączeniem standardu języka Verilog i jego rozszerzeń rozgorzały na nowo. Ostatecznie IEEE zdecydował o dokonaniu takiego połączenia. Prace zmierzające w tym kierunku rozpoczęły się już w pierwszych miesiącach 2007 roku. Jednocześnie rozwiązano grupę roboczą pracującą nad standardem IEEE 1364. Wszystkie prace prowadzone przez tę grupę skierowane do grupy IEEE 1800 System Verilog. Fakt ten oznaczał, że dotychczasowa nazwa Verilog nie miała być już dłużej stosowana, chyba że



Zestawienie najważniejszych cech czterech generacji języka Verilog

w odniesieniu do wcześniejszych wersji języka. Nową nazwą wspólnego standardu miał zostać System Verilog. Zmierzch nazwy „Verilog" nie oznacza jednak końca metodologii projektowania opartej na języku Verilog. Sugeruje zaś, że Verilog jest językiem wciąż intensywnie rozwijanym i ulepszanym.

Na ilustracji przedstawiono w sposób symboliczny najważniejsze cechy poszczególnych standardów języka Verilog, począwszy od pierwszej wersji języka, mającej swoje korzenie jeszcze w roku 1984, aż do 4. generacji – System Verilog-2005. Pod koniec 2008 lub na początku 2009 roku spodziewane było ogłoszenie najnowszego standardu (5. generacji języka), będącego wynikiem pracy grupy roboczej IEEE 1800 System Verilog. Jednak w chwili pisania tego tekstu oficjalnej decyzji dotyczącej tego faktu jeszcze nie ogłoszono.

Na podstawie podanego rysu historycznego mogliśmy się przekonać, że Verilog jest językiem przechodzącym w ostatnich latach sporą ewolucję. Jednak praktycznie wszystkie popularne systemy projektowe przeznaczone dla układów FPGA i CPLD, takie jak Xilinx ISE czy Altera Quartus, nawet w swoich najnowszych wersjach, oferują do dyspozycji użytkowników, oprócz VHDLa, język Verilog standardu IEEE 1364-2001. Z tego powodu, w oddawanym w ręce Czytelników opracowaniu zajmiemy się bliżej tym właśnie standardem. W poszczególnych rozdziałach omówimy wszystkie zasadnicze elementy języka, koncentrując się przede wszystkim na zastosowaniach Veriloga do celów syntezy logicznej. Aczkolwiek nie pominiemy również wielu zagadnień związanych z symulacją.

Ze względu na wielość poszczególnych aspektów dotyczących samego języka, niektóre zagadnienia omówimy w formie skróconej, kilka z nich jedynie zasygnalizujemy, a kilka mniej istotnych z punktu widzenia syntezy – pominiemy. Przykładowo nie będziemy szerzej omawiać tematu podstawowych elementów logicznych definiowanych przez użytkownika (UDP), bloku *specify*, projektowania z dokładnością do cyklu (*cycle accurate design*), zaawansowanych zagadnień odnoszących się do przyjętego modelu czasowego dla symulacji czy też modelowania na poziomie przełączników (*switch level modelling*). Z tego też powodu, w tytule książki użyliśmy terminu „wprowadzenie", sugerując niejako, że omawiany będzie pewien, mimo wszystko dość obszerny, podzbiór szerszego zbioru zagadnień związanych z językiem Verilog i jego najnowszą wersją – System Verilog.

Materiał zawarty w książce podzielono na pięć rozdziałów i dwa dodatki. Rozdział pierwszy koncentruje się wokół tematyki projektowania strukturalnego, a także operatorów języka Verilog. Omawiane są takie zagadnienia jak typy danych, moduł, instancje, tablice instancji, hierarchia, selekcja bitu i częściowa selekcja, podstawowe elementy logiczne, przypisania ciągłe oraz – szczegółowo – wszystkie występujące w języku Verilog operatory.

Rozdział drugi ma charakter krótkiego wprowadzenia w tematykę symulacji. W szczególności przedstawiane w rozdziale treści to przygotowanie symulacji, jednostka testowa, instrukcje *always* i *initial*, instrukcja opóźnienia, blok *begin...end* oraz *fork...join*, instrukcje formatowania i wypisywania tekstu wyników symulacji, obsługa plików.

Rozdziały trzeci i czwarty dotyczą modelowania behawioralnego. W rozdziale trzecim omawiane są zagadnienia behawioralnego modelowania układów kombinacyj-

nych, takie jak: reguły specyfikacji układów kombinacyjnych, lista wrażliwości procesu, instrukcja warunkowa, wnioskowanie rejestrów zatrzaskowych, instrukcje *case*, *disable* i *generate*, pętle, funkcje i zadania, zasięg nazw oraz nazwy hierarchiczne. Rozdział czwarty przedstawia tematykę behawioralnego modelowania układów kombinacyjnych, w tym: elementy sekwencyjne – zatrzaski i przerzutniki *filp-flop*, reguły specyfikacji układów sekwencyjnych, przypisania proceduralne: blokujące, nieblokujące oraz proceduralne przypisania ciągłe, modelowanie automatów sekwencyjnych, modelowanie procesów współbieżnych, instrukcja zdarzenia oraz instrukcja *wait*.

W rozdziale piątym zaprezentowano wiele przykładowych projektów ilustrujących w sposób praktyczny zagadnienia omawiane w poprzednich rozdziałach. Można tu znaleźć zarówno stosunkowo proste przykłady, takie jak sterowniki 7-segmentowego wyświetlacza LED oraz alfanumerycznego wyświetlacza LCD, konwertery kodów binarno-dziesiętnych, podstawowe układy arytmetyczne czy też port szeregowy z buforem FIFO, jak również projekty nieco bardziej złożone. Do tych ostatnich można zaliczyć moduł obsługi magistrali 1-Wire oraz opisy behawioralne dwóch 8-bitowych mikrokontrolerów: znanego z opisu strukturalnego mikrokontrolera PicoBlaze oraz mikrokontrolera opartego na rodzinie PIC16F8xx *Microchip*. Obydwa opracowane mikrokontrolery charakteryzują się pewnymi ulepszeniami architektury, zwiększającymi ich ogólną wydajność (zmniejszenie liczby taktów zegara przypadających na jeden cykl maszynowy) w stosunku do pierwowzorów. Pokazany jest również sposób integracji tych mikrokontrolerów z omawianymi wcześniej modułami logicznymi portu szeregowego, konwerterów kodu, sterownika alfanumerycznego wyświetlacza LCD oraz obsługi magistrali jednoprzewodowej 1-Wire. Dodatkowo dla obydwu opracowanych mikrokontrolerów pokazane są sprzętowe moduły pełniące podobną funkcję jak tzw. *bootloadery* dla mikrokontrolerów ASIC. Odgrywają one użyteczną rolę podczas uruchamiania oprogramowania dla mikrokontrolerów, pozwalając w prosty sposób załadować kod wykonywalny do pamięci programu za pomocą portu szeregowego oraz, na przykład, aplikacji *HyperTerminal*. Moduły te interpretują pliki z kodem wynikowym w formacie Intel HEX 8.

Materiał prezentowany w książce uzupełniają dwa dodatki. W pierwszym z nich znajduje się formalny opis składni języka Verilog standardu IEEE 1364-2001. Drugi dodatek skrótowo opisuje interesujące zagadnienia związane z realizacją układów asynchronicznych w kontekście ich implementacji w strukturach FPGA. W dodatku można znaleźć podstawową charakterystykę poszczególnych kategorii, na jakie dzielą się układy asynchroniczne, w tym klasycznych układów typu Huffmana, układów z samotaktowaniem (*self-clocked*) oraz ze współpracą lokalną (*self-timed*), a także układów mikropotokowych. Omawiana jest również tematyka hazardu w układach kombinacyjnych i sekwencyjnych (hazard funkcyjny, logiczny, zasadniczy), szczególnie w kontekście implementacji tych układów w strukturach FPGA. Dodatek kończy informacja o zjawisku metastabilności w układach cyfrowych.