

**Bogdan Pankiewicz
Marek Wójcikowski**

Języki modelowania i symulacji

Gdańsk 2017

PRZEWODNICZĄCY KOMITETU REDAKCYJNEGO
WYDAWNICTWA POLITECHNIKI GDAŃSKIEJ
Janusz T. Cieśliński

RECENZENT
Stanisław Szczepański

PROJEKT OKŁADKI
Wioleta Lipska-Kamińska

Wydanie I – 2015

Wydano za zgodą
Rektora Politechniki Gdańskiej

Oferta wydawnicza Politechniki Gdańskiej jest dostępna pod adresem
<http://www.pg.edu.pl/wydawnictwo/katalog>
zamówienia prosimy kierować na adres wydaw@pg.edu.pl

Utwór nie może być powielany i rozpowszechniany, w jakiegokolwiek formie
i w jakiegokolwiek sposób, bez pisemnej zgody wydawcy

© Copyright by Wydawnictwo Politechniki Gdańskiej
Gdańsk 2017

ISBN 978-83-7348-715-4

WYDAWNICTWO POLITECHNIKI GDAŃSKIEJ

Wydanie II. Ark. wyd. 7,3, ark. druku 7,5, 1167/999

Druk i oprawa: Volumina.pl Daniel Krzanowski
ul. Księcia Witolda 7-9, 71-063 Szczecin, tel. 91 812 09 08

Spis treści

1. Wstęp	7
Wykaz ważniejszych skrótów	8
2. Symulacje układów elektronicznych z wykorzystaniem PSPICE	9
2.1. Informacje wstępne	9
2.2. Zasady ogólne składni PSPICE	9
2.3. Jednostki i zasady zapisu wartości	11
2.4. Temperatura w PSPICE	12
2.5. Modele elementów	12
2.6. Elementy z dwoma wyprowadzeniami	13
2.6.1. Rezystor	14
2.6.2. Kondensator	15
2.6.3. Cewka indukcyjna	16
2.6.4. Niezależne źródło napięciowe i prądowe	16
2.7. Podstawowe rodzaje analiz oraz sterowanie wyjściami	20
2.7.1. Analiza stałoprądowa	22
2.7.2. Analiza częstotliwościowa małosygnałowa	25
2.7.3. Analiza czasowa	27
2.8. Analizy pochodne	30
2.8.1. Analiza Fouriera	30
2.8.2. Analiza .TF	32
2.8.3. Analiza wrażliwościowa	32
2.8.4. Analiza szumowa	34
2.9. Ustalenie punktu pracy i przybliżonego punktu pracy	36
2.9.1. Polecenie .IC	36
2.9.2. Polecenie .NODESET	36
2.10. Operacje na plikach	37
2.10.1. Polecenie .INC	37
2.10.2. Polecenie .LIB	37
2.10.3. Polecenie .SAVEBIAS	37
2.10.4. Polecenie .LOADBIAS	38
2.11. Wybrane elementy półprzewodnikowe	38
2.11.1. Dioda półprzewodnikowa	38
2.11.2. Tranzystor bipolarny	39
2.11.3. Tranzystor MOS	39
2.12. Źródła sterowane napięciem	40
2.13. Źródła sterowane prądem	41
2.14. Podukłady, deklaracja i wstawienie	41
2.14.1. Deklaracja podukładu	41
2.14.2. Wstawienie podukładu	42
2.15. Deklaracja parametru	45
2.16. Analiza parametryczna	45
2.17. Operatory i funkcje wbudowane oraz deklaracja funkcji własnych	47
2.18. Analiza Monte Carlo	48
3. Wstęp do języków HDL	53
3.1. Symulacje i testowanie	54

4. Język Verilog	55
4.1. Pojęcia podstawowe	55
4.1.1. Podstawowe zasady składni języka Verilog	55
4.1.2. Operatory	55
4.1.3. Liczby	56
4.1.4. Identyfikatory	56
4.1.5. Zestaw wartości	57
4.1.6. Sieci	57
4.1.7. Rejestry	57
4.1.8. Wektory	58
4.1.9. Liczby całkowite i rzeczywiste	58
4.1.10. Tablice i pamięci	58
4.1.11. Łańcuchy	59
4.1.12. Zadania systemowe	59
4.1.13. Dyrektywy kompilatora	60
4.2. Moduły i porty	61
4.2.1. Porty	62
4.2.2. Definiowanie parametrów modułu	63
4.3. Projektowanie i modelowanie na poziomie bramek logicznych	64
4.3.1. Bramki	64
4.3.2. Opóźnienia w bramkach	65
4.4. Projektowanie i modelowanie na poziomie rejestrów	67
4.4.1. Przypisanie ciągle	67
4.4.2. Opóźnienia	68
4.4.3. Wyrażenia i operatory	68
4.5. Projektowanie i modelowanie na poziomie behawioralnym	72
4.5.1. Procedury strukturalne	72
4.5.2. Przypisanie proceduralne	72
4.5.3. Sterowanie wykonaniem instrukcji	74
4.5.4. Różnica pomiędzy przypisaniem <i>blocking</i> i <i>nonblocking</i>	74
4.5.5. Wyrażenie warunkowe <i>if</i>	75
4.5.6. Wyrażenie typu <i>case</i>	75
4.5.7. Pętle	76
4.5.8. Bloki sekwencyjne i równoległe	77
4.6. Zadania i funkcje	77
4.7. Techniki modelowania	78
4.7.1. Proceduralne przypisanie ciągle	78
4.7.2. Skala czasu	79
4.7.3. Praca z plikami	79
4.8. Verilog 2001	79
4.8.1. Blok konfiguracji	80
4.8.2. Polecenie <i>generate</i>	80
4.8.3. Nowy sposób indeksowania wektorów	80
4.8.4. Tablice wielowymiarowe	80
4.8.5. Operacje na liczbach <i>signed</i>	81
4.8.6. Inne zmiany	81
5. Język VHDL	82
5.1. Pojęcia podstawowe	82
5.1.1. Podstawowe zasady składni języka VHDL	83
5.1.2. Identyfikatory	83
5.1.3. Literały	85
5.1.4. Typ wyliczeniowy	85
5.1.5. Typ całkowity	86
5.1.6. Typy tablicowe	86

5.1.7. Typ łańcuchowy <i>string</i>	89
5.1.8. Typ <i>bit_vector</i>	89
5.1.9. Rekordy	89
5.1.10. Typ rzeczywisty	90
5.1.11. Typ fizyczny	90
5.1.12. Typy predefiniowane	90
5.1.13. Podtypy	90
5.1.14. <i>Aliases</i>	90
5.1.15. Konwersja typów	91
5.1.16. Podsumowanie najważniejszych typów danych	91
5.1.17. Biblioteki	91
5.1.18. Pakiet <i>std_logic_1164</i>	92
5.1.19. Pakiet <i>std_logic_arith</i>	93
5.1.20. Pakiet <i>std_logic_unsigned</i>	95
5.1.21. Pakiet <i>std_logic_signed</i>	95
5.1.22. Tworzenie własnego pakietu	95
5.2. Poziom strukturalny	96
5.2.1. Blok <i>entity</i>	96
5.2.2. Blok architektury	96
5.2.3. Stałe	97
5.2.4. Sygnały	97
5.2.5. Osadzanie komponentu	98
5.2.6. Polecenie <i>generate</i>	99
5.2.7. Parametry bloku <i>entity (generic)</i>	100
5.3. Poziom przesłań międzyrejstrowych RTL	101
5.3.1. Przypisanie współbieżne	101
5.3.2. Współbieżne przypisanie warunkowe <i>when ... else</i>	102
5.3.3. Przypisanie współbieżne <i>select ... when</i>	102
5.3.4. Różnice pomiędzy przypisaniem <i>when ... else</i> i <i>select ... when</i>	103
5.3.5. Operatory logiczne	103
5.3.6. Operatory porównania	103
5.3.7. Operatory dodawania i konkatencji	103
5.3.8. Inne operatory	104
5.3.9. Opóźnienia	104
5.3.10. Instrukcje współbieżne i sekwencyjne	105
5.3.11. Procesy	105
5.3.12. Zmienne	106
5.3.13. Przypisanie sekwencyjne	106
5.3.14. Różnice pomiędzy sygnałem i zmienną	106
5.4. Abstrakcyjny poziom behawioralny	107
5.4.1. Wyrażenie warunkowe <i>if</i>	107
5.4.2. Wyrażenie warunkowe <i>case</i>	108
5.4.3. Polecenia pętli <i>loop</i>	108
5.4.4. Polecenie <i>next</i>	111
5.4.5. Polecenie <i>exit</i>	111
5.4.6. Podstawowe rodzaje procesów	111
5.4.7. Opóźnienia typu <i>wait</i>	113
5.5. Funkcje i procedury	114
5.5.1. Funkcje	114
5.5.2. Procedury	115
6. Podsumowanie	117
Literatura	118

1. Wstęp

Szybki rozwój technologii wytwarzania elementów i układów scalonych umożliwia tworzenie coraz bardziej złożonych urządzeń i systemów elektronicznych. Niestety, ta powiększająca się złożoność stanowi istotny problem i wyzwanie w procesie projektowania urządzenia. Pojedynczej osobie czy grupie projektowej, nawet bardzo dobrze zaznajomionej z bieżącym stanem techniki, trudno jest przewidzieć wszystkie możliwe zachowania złożonego systemu elektronicznego. Wszelkie ewentualnie popełnione błędy powodują konieczność wykonania poprawek, co stanowi problem w odniesieniu do szybkiego wejścia produktu na rynek, a także zwiększa koszty ogólne projektu. Z tego względu od momentu pojawienia się pierwszych komputerów podejmuje się próby symulacji rzeczywistych układów elektronicznych w sztucznym wirtualnym środowisku obliczeniowym. Takie symulacje mają pozwolić odpowiedzieć na pytanie, jak zachowa się badany obwód elektryczny w rzeczywistości na podstawie jego opisu za pomocą modeli matematycznych (obliczeniowych). Jednym z pierwszych symulatorów elektrycznych jest program SPICE (ang. *Simulation Program with Integrated Circuit Emphasis*), który po przeniesieniu na platformę komputerów typu PC został nazwany PSPICE. Program ten to symulator operujący w dziedzinie elektrycznej i umożliwiający symulację wszelkiego rodzaju obwodów elektrycznych składających się z elementów, których modele są dostępne w programie PSPICE. Potrzeba powstania takiego symulatora była szczególnie istotna przy projektowaniu układów scalonych, gdzie koszty uruchomienia produkcji są wysokie i dlatego zachodzi potrzeba, aby pierwsza wersja projektu była pozbawiona wszelkich błędów i nie wymagała wprowadzania późniejszych poprawek. Symulatory elektryczne są do dzisiaj bardzo często stosowane przy projektowaniu układów zarówno analogowych, jak i cyfrowych. Powstało wiele odmian takiego oprogramowania, z których najbardziej znane to: PSPICE, HSPICE, Spectre, APS, LTSpice i inne. Dodatkowo, ze względu na dużą liczbę elementów tworzących dzisiejsze systemy cyfrowe powstały symulatory logiczne, które kosztem zmniejszenia precyzji obliczeniowej umożliwiają symulację bardzo złożonych układów cyfrowych. W tych symulatorach nie występuje pojęcie wartości napięcia czy prądu; zamiast tego mamy sygnały w postaci stanu logicznego (np. wysoki, niski lub wysokiej impedancji), których propagacja od wejścia bloku cyfrowego do jego wyjścia zajmuje pewien czas zdefiniowany w modelu danego bloku. Takie uproszczenie umożliwia znaczące przyspieszenie symulacji, zostaje jednak przy tym utracona część informacji, nieistotnych w pierwszym przybliżeniu symulacji układu cyfrowego, takich jak np. pobór mocy, przesłuchy międzysygnałowe, szумы, zakłócenia i wiele innych. Symulatory logiczne ewaluowały przez dłuższy okres i obecnie można przyjąć, że znaczącej większości z nich można użyć poprzez zastosowanie dwóch rodzajów języków opisu sprzętu cyfrowego typu HDL (ang. *Hardware Description Language*): Verilog i VHDL (ang. *Very High Speed Integrated Circuit Hardware Description Language*). Języki te są bardzo rozpowszechnione zwłaszcza wśród projektantów układów wykorzystujących programowalne układy cyfrowe typu CPLD (ang. *Complex Programmable Logic Device*) lub FPGA (ang. *Field Programmable Gate Array*), za pomocą których oprócz symulacji można również wykonać projekt bloku cyfrowego. Ostatnio w wielu systemach znajdują zastosowanie układy mieszane, które pracują zarówno z sygnałami cyfrowymi, jak i analogowymi. Dla takiego rodzaju układów nie możemy niestety wykorzystać typowych języków HDL gdyż w ten sposób zostałaby utracona informacja analogowa w postaci napięć i prądów występujących w obwodzie. Takie układy możemy symulować za pomocą symulatorów elektrycznych, wówczas jednak wymagany jest bardzo długi czas na obliczenia, gdyż część cyfrowa, zazwyczaj stanowiąca znaczącą większość systemu mieszanego, jest symulowana bez uproszczeń. Z tego względu pojawiły się języki opisu systemów mieszanych, takie jak np. Verilog-A (ang. *Verilog-Analog*) czy VHDL-AMS (ang. *VHDL – Analog Mixed Signal*), które do opisu wykorzystują jednocześnie dziedzinę elektryczną i logiczną.

W niniejszym skrypcie przedstawiono podstawy wykorzystania symulatora PSPICE, języka Verilog oraz języka VHDL. Opracowanie składa się z trzech głównych części omawiających kolejno poszczególne zagadnienia. Szczególny nacisk położono na zasady wykorzystania i nabycie praktycznych umiejętności dotyczących symulacji układów elektronicznych. Zainteresowanych metodami numerycznymi wykorzystywanymi w symulatorach zachęcamy do skorzystania z dodatkowej literatury specjalistycznej [1–10].

Wykaz ważniejszych skrótów

ASIC	układ scalony zaprojektowany do szczególnego, specyficznego zastosowania (ang. <i>Application Specific Integrated Circuit</i>)
CAD	zastosowanie sprzętu komputerowego i oprogramowania w celu wspomaganie projektowania technicznego w wielu dziedzinach, m.in. w elektronice, mechanice i inżynierii budowlanej (ang. <i>Computer Aided Design</i>)
CPLD	złożony, programowalny układ cyfrowy, którego budowa opiera się na makrokomórkach składających się z programowalnych pól iloczynów i sum logicznych oraz elementów wyjściowych (ang. <i>Complex Programmable Logic Device</i>)
FPGA	bardzo rozbudowany cyfrowy układ programowalny składający się z rozmieszczonych matrycowo programowalnych bloków logicznych (ang. <i>Field Programmable Gate Array</i>)
IEEE	instytut inżynierów elektryków i elektroników, instytucja typu <i>non-profit</i> , której jednym z podstawowych zadań jest ustalanie standardów dla urządzeń elektronicznych i komputerowych (ang. <i>Institute of Electrical and Electronics Engineers</i>)
MVL4	logika bazująca na czterech wartościach poziomów logicznych (ang. <i>Multi-Valued Logic</i>)
PSPICE	oprogramowanie na komputery klasy PC do symulacji obwodów elektrycznych ze szczególnym uwzględnieniem potrzeb projektowania układów scalonych (ang. <i>Personal Computer Simulation Program with Integrated Circuit Emphasis</i>)
RTL	poziom opis układów cyfrowych bazujący na przesłaniach międzyrejestrowych (ang. <i>Register Transfer Level</i>)
SI	międzynarodowy układ jednostek i miar (fr. <i>Système international d'unités</i>)
SPICE	oprogramowanie do symulacji obwodów elektrycznych ze szczególnym uwzględnieniem potrzeb projektowania układów scalonych (ang. <i>Simulation Program with Integrated Circuit Emphasis</i>)
THD	całkowita zawartość harmonicznych (ang. <i>Total Harmonic Distortion</i>)
UUT	jednostka poddana testowi (ang. <i>Unit Under Test</i>)
VHDL	język opisu sprzętu dla cyfrowych układów scalonych o dużej szybkości pracy (ang. <i>Very High Speed Integrated Circuit Hardware Description Language</i>)
VHDL-AMS	język VHDL uzupełniony o możliwość opisu układów scalonych analogowych i mieszanych (ang. <i>VHDL-Analog Mixed Signal</i>)
VHPI	interfejs językowy do języka VHDL (ang. <i>VHDL Procedural Interface</i>)
VLSI	bardzo duża skala integracji (ang. <i>Very Large Scale of Integration</i>)