Przykładowe pytania z części PSPICE.

1. Podaj zasady tworzenia pliku symulacyjnego.
2. Czy składnia PSPICE jest czuła na wielkość liter?
3. Jak umieszcza się komentarze w pliku symulacyjnym PSPICE?
4. Jak definiuje się rezystor?
5. Jak definiuje się kondensator?
6. Jak definiuje się cewkę?
7. Jak definiuje się niezależne źródło napięciowe i prądowe?
8. Jak oznaczony jest globalny węzeł masy?
9. Co to są węzły „floating" i czy są dozwolone w symulowanym obwodzie?
10. Jakie przyrostki wartości można stosować w składni SPICE?
11. Jaka jest dopuszczalna składnia zapisu wartości?
12. Co to jest temperatura nominalna i bieżąca temperatura symulacji?
13. Jakie są zasady w definiowaniu modeli?
14. Jaki jest model rezystora?
15. Jaki jest model cewki?
16. Jaki jest model kondensatora?
17. Jakie są analizy podstawowe możliwe do wykonania w symulatorze PSPICE?
18. Co to jest analiza stałoprądowa?
19. Co to jest analiza częstotliwościowa?
20. Co to jest analiza czasowa?
21. Co to jest analiza Fouriera?
22. Co to jest analiza .OP?
23. Co to jest analiza .TF?
24. Co to jest analiza .SENS?
25. Co to jest analiza .NOISE?
26. Gdzie umieszczane są wyniki analiz wykonywanych w PSPICE?
27. Do czego służy polecenie .IC?
28. Do czego służy polecenie .NODESET?
29. Do czego służy polecenie .SAVEBIAS?
30. Do czego służy polecenie .LOADBIAS?
31. Jakie polecenia służą do operacji na plikach i jakie są skutki ich działania?
32. Jak powołuje się tranzystor bipolarny?
33. Jak powołuje się tranzystor MOS?
34. Jak powołuje się diodę?
35. Jak powołuje się napięciowe i prądowe źródła sterowane?
36. Jak deklaruje się i powołuje podukłady?
37. Jak można się odnieść do elementów, prądów i napięć w podukładach?
38. Do czego służy polecenie .PARAM?
39. Do czego służy polecenie .FUNC?
40. Podaj listę operatorów i funkcji predefiniowanych w PSPICE.
41. Do czego służy polecenie .STEP?
42. Do czego służy polecenie .TEMP?

Przykładowe pytania z języka Verilog. Jako odpowiedź należy podać czy dane zdanie stanowi prawdę (P) czy fałsz (F).

1. Identyfikator musi zaczynać się literą lub cyfrą.
2. Każdy system cyfrowy w Verilog'u jest reprezentowany poprzez moduł (*module*).
3. Każde wyprowadzenie modułu (port) musi być zadeklarowane poniżej listy portów (w standardzie '95).
4. Identyfikatory „Bus" oraz „BUS" są w języku Verilog różnymi identyfikatorami.
5. Przypisanie ciągłe *assign* określa wartość wyjścia (lewego operanda) w zależności od wartości wejść i operacji na nich.
6. Opis typu behawioralnego wykonywany jest wewnątrz bloków *always* oraz *initial*.
7. Każdy moduł (*module*) może być wstawiony jako komponent wewnątrz innego modułu.
8. Możliwe jest wykonanie komentarzy obejmujących tylko jedną linię.
9. Każda sieć (*net*) może przyjąć jedną z następujących 4 wartości: '0', '1', 'Z', 'X'.
10. Stan nieznany 'X' oraz wysokiej impedancji 'Z' ma takie samo znaczenie dla operacji logicznych.
11. Jeśli jeden z operandów operatora logicznego ma stan nieznany 'X' wówczas wynik również ma wartość nieznaną 'X'.
12. Sieci deklarowane są wyrażeniem *net*.
13. Każdy sygnał musi być deklarowany indywidualnie.
14. Bit najbardziej na lewo wektora jest zawsze najbardziej znaczący.
15. Deklaracja portów modułu musi być zawsze pierwszym składnikiem wnętrza modułu (std. '95).
16. Wszystkie porty są domyślnie sieciami.
17. Każdy port może być typu *reg*.
18. Jeśli port wyjściowy jest typu *reg* wówczas odczyt jego wartości przez moduł nie jest możliwy.
19. Indeks najbardziej znaczącego bitu wektora może być mniejszy niż najmniej znaczącego.
20. Pierwszy port predefiniowanych bramek logicznych jest zawsze wejściem.
21. Predefiniowana bramka logiczna *and* może mieć dowolna liczbę wejść.
22. Kolejność wstawień poszczególnych bramek predefiniowanych nie ma znaczenia dla działania opisywanego układu.
23. Wszystkie porty wstawianego komponentu (modułu) muszą być podłączone.
24. Możliwe jest łączenie portów wstawianych modułów poprzez uporządkowaną listę jak również poprzez nazwy portów.
25. Wynikiem działania operatora relacyjnego jest wartość TRUE lub FALSE.
26. Operatory logiczne dają jako wynik zawsze jeden bit.
27. W operatorze przesuwania (<<) puste bity są zawsze uzupełniane '0' lub '1' w zależności od pierwszego bitu wychodzącego.
28. Jeśli jeden z operandów jest równy 'X' wówczas wynik porównania przy użyciu operatora '= =' też będzie równy 'X'.
29. Przypisania ciągłe (*assign*) mogą być stosowane tylko dla określania wartości sygnałów typu sieć (*net*).
30. Kolejność występowania przypisań ciągłych ma istotne znaczenie dla działania opisywanego systemu cyfrowego.
31. Opóźnienia w Verilogu są podawane w nanosekundach.
32. Przypisanie warunkowe (*? :*) jest operatorem trójargumentowym.
33. Deklaracja sygnału jako typu *reg* daje po syntezie zawsze sygnał wyjściowy przerzutnika.
34. Deklaracja tablicy składa się z identyfikatora po którym występuje zakres tablicy.
35. *Parameter* jest w Verilog'u odpowiednikiem stałej.
36. Wszystkie wyrażenia typu behawioralnego muszą znajdować się wewnątrz bloków *initial* lub *always*.
37. Lewostronnym operandem przypisania wartości w bloku behawioralnym mogą być zarówno sieci jak i rejestry.
38. Nie jest możliwy bezpośredni dostęp do pojedynczego bitu tablicy.
39. Każdy blok *initial* jest wykonywany jednokrotnie.
40. Argument wyrażenia *posedge* musi być zawarty w nawiasach np. *posedge(clk)*.
41. Przypisanie typu nonblocking (<=) umożliwia opis zdarzeń występujących współbieżnie.
42. Sygnały na liście zdarzeń są rozdzielone przecinkami (std. '95).
43. Funkcja może zawierać opóźnienie, zadanie nie może.
44. Zadanie może mieć dowolną liczbę argumentów.
45. Blok *always* opisujący układ sekwencyjny musi mieć na liście zdarzeń wszystkie sygnały wejściowe układu oraz sygnał zegarowy.
46. Wszystkie zadania systemowe mają identyfikator rozpoczynający się od znaku '$'.
47. Blok *always* opisujący układ kombinacyjny musi mieć na swojej liście zdarzeń umieszczone wszystkie sygnały wejściowe (std. '95).

48. Funkcje mogą mieć argumenty wyjściowe.
49. Testowany moduł jest zazwyczaj wstawiany jako komponent do modułu TEST BENCH.

Przykładowe pytania z języka VHDL. Jako odpowiedź należy podać czy dane zdanie stanowi prawdę (P) czy fałsz (F).

1. Deklaracja ENTITY może zawierać tylko nazwę bloku oraz jego wyprowadzenia WE/WY.
2. Opis na poziomie strukturalnym może być hierarchiczny.
3. Wejścia do bloku ENTITY nazywane są GENERIC a wyjścia PORT.
4. Jeden blok ENTITY może mieć wiele bloków ARCHITECTURE.
5. Każdy blok ENTITY musi mieć listę PORTÓw.
6. Aby używać zawartości pakietu STANDARD należy przed deklaracją ENTITY umieścić klauzule LIBRARY oraz USE.
7. System cyfrowy opisany w języku VHDL składa się z deklaracji ENTITY oraz ARCHITECTURE.
8. Jeden blok ARCHITECTURE może być przypisany do wielu bloków ENTITY.
9. Szyna danych o szerokości 8-miu bitów jest reprezentowana jako typ BYTE.
10. Sygnały które łączą system z otoczeniem nazywane są PORTami.
11. Szerokość szyny danych jest zdefiniowana w czasie deklaracji sygnału.
12. Kolejność bitów w szynie danych nie ma znaczenia.
13. Sygnały wewnętrzne systemu są deklarowane wewnątrz bloku ARCHITECTURE.
14. Lewy indeks w deklaracji szyny danych musi zawsze być większy niż prawy.
15. Każdy PORT musi być podany wraz z kierunkiem przesyłania sygnału.
16. Wszystkie sygnały są deklarowane w bloku ENTITY.
17. Nazwa bloku ENTITY może pojawić się za wyrażeniem END ENTITY;
18. PORT musi być albo trybu IN albo OUT.
19. GENERICs mogą być dynamicznie zmieniane w czasie symulacji kodu VHDL.
20. Komentarze zaczynają się i kończą dwoma myślnikami --.
21. PORTy są statycznymi połączeniami bloku ENTITY z otoczeniem i nie zmieniają swojej wartości w czasie symulacji.
22. GENERICs umożliwiają wprowadzenie stałych informacji do wnętrza bloku ENTITY.
23. VHDL jest językiem czułym na wielkość liter.
24. GENERICs mogą być użyte do specyfikacji PORTów np. ich długości.
25. Żaden identyfikator nie może zawierać spacji.
26. Każdy GENERIC musi być zadeklarowany łącznie z jego trybem (kierunkiem).
27. PORTy są sygnałami.
28. Jest dozwolone podawanie wartości początkowej PORTu.
29. Wartość całego wektora musi być wpisana w podwójnym cudzysłowiu podczas gdy jego części w pojedynczym.
30. Znak przypisania wartości sygnału może być wpisany jako <= lub => w zależności od potrzeb projektanta.
31. Stałe mogą być używane w wyrażeniach.
32. Zestaw stanów maszyny stanów jest zazwyczaj deklarowany jako typ wyliczeniowy.
33. Elementy tablicy muszą być tego samego typu.
34. Stałej może być przypisana nowa wartość jeśli jest taka sama jak poprzednia.
35. Wartość TRUE jest ekwiwalentem wartości ‘1’.
36. Operatory są zawsze zdeklarowane dla danego typu wartości.
37. Operatory logiczne mogą być stosowane tylko dla pojedynczych bitów.
38. Ponieważ PROCESS jest nieskończoną pętlą więc wyrażenie WAIT umieszczone na początku lub końcu procesu daje takie same wyniki.
39. Wykonywanie PROCESSu kończy się w momencie osiągnięcia wyrażenia END PROCESS.
40. PROCESS z listą czułości nie może zawierać wyrażenia WAIT.
41. Wartości sygnałów i zmiennych tego samego typu mogą być wzajemnie przypisywane.
42. PROCESS jest zestawem instrukcji sekwencyjnych.
43. Po słowie PROCESS można zdeklarować jego nazwę (inaczej etykietę).
44. Wyrażenie WAIT umieszczone w procesie specyfikuje warunek zawieszenia procesu.
45. Sygnały typu STD_LOGIC oraz STD_LOGIC_VECTOR są przemysłowym standardem sygnałów, do których mogą być przypisane wielokrotne źródła sygnałów gdyż posiadają tzw. funkcję arbitrażową.
46. Sygnał w procesie ma tylko jeden generator wartości niezależnie jak wiele przypisań wartości do niego w procesie występuje.
47. Inaczej niż w przypadku innych składników języka VHDL, etykiety komponentów są obowiązkowe.
48. PORTY komponentów które nie są używane muszą być przypisane do wyrażenia OPEN i nie mogą być pominięte.
49. GENERICom podanym w czasie deklaracji komponentu mogą być przypisane różne wartości w różnych wstawieniach tych komponentów.
50. Komponenty są połączone tylko poprzez sygnały zadeklarowane wewnątrz bloku ARCHITECTURE.
51. W przypadki mapowania pozycyjnego sygnały są podane w tej samej kolejności jak w deklaracji bloku ENTITY.
52. W przypadku mapowania poprzez nazwę używa się znaku <= lub => w zależności od trybu portu (tj. IN lub OUT).
53. Testowany blok ENTITY jest zazwyczaj symulowany jako komponent wewnątrz Test_Bench.
54. Blok Test_Bench nie zawiera PORTów.
55. Generacja sygnałów testowych musi następować bezpośrednio w bloku Test_Bench.
56. Blok Test_Bench nie zawiera PORTów i dlatego deklaracja ENTITY tego bloku nie jest potrzebna.

Odpowiedzi Verilog:

1F, 2P, 3P, 4P, 5P , 6P, 7P, 8P, 9P, 10P, 11F, 12F, 13F, 14P, 15P, 16P, 17F, 18F, 19P, 20F, 21P, 22P, 23F, 24P, 25F, 26P, 27F, 28P, 29P, 30F, 31F, 32P, 33F, 34P, 35P, 36P, 37F, 38P, 39P, 40F, 41P, 42F, 43F, 44P, 45F, 46P, 47P, 48F, 49P

Odpowiedzi VHDL:

1F, 2P, 3F, 4P, 5F , 6F, 7P, 8F, 9F, 10P, 11P, 12F, 13P, 14F, 15F, 16F, 17P, 18F, 19F, 20F, 21F, 22P, 23F, 24P, 25P, 26F, 27P, 28P, 29F, 30F, 31P, 32P, 33P, 34F, 35F, 36P, 37F, 38F, 39F, 40P, 41P, 42P, 43F, 44F, 45P, 46P, 47P, 48F, 49P, 50F, 51P, 52F, 53P, 54P, 55F, 56F