**Это пиздец товарищи**

1. Методы оценки погрешностей математических операций: аналитический, численный и статистический.
2. Погрешности при выполнении математических операций в двоичном коде наблюдаются из-за ограничения разрядности и представления чисел в двоичной системе счисления.
3. Источники погрешностей при выполнении операции умножения в двоичном коде: округление, переполнение и ошибки при представлении чисел.
4. Источники погрешностей при выполнении операции деления в двоичном коде: округление, переполнение и ошибки при представлении чисел.
5. Общие правила проведения сложений в двоичном коде: сложение по модулю 2, перенос при необходимости.
6. Общие правила проведения вычитания в двоичном коде: вычитание по модулю 2, заем при необходимости.
7. Общие правила выполнения умножения чисел в двоичном коде: умножение по модулю 2, сдвиги и сложения.
8. Операции деления и умножения производят обычно в формате с плавающей точкой, чтобы обеспечить большую точность и диапазон представления чисел.
9. Общие правила деления двоичных чисел: деление по модулю 2, сдвиги и сравнения.
10. Количество итераций при делении двоичных чисел зависит от размера чисел и алгоритма деления.
11. Алгоритм деления с восстановлением остатка: деление по модулю 2, сдвиги и сравнения, с сохранением остатка.
12. Преимущества деления с восстановлением остатка: точность и сохранение информации. Недостатки: сложность и время выполнения.
13. Алгоритм деления без восстановления остатка: деление по модулю 2, сдвиги и сравнения, без сохранения остатка.
14. Преимущества деления без восстановления остатка: простота и скорость. Недостатки: потеря информации и ошибки.
15. Методы деления двоичных чисел: деление по модулю 2, сдвиги и сравнения, деление с восстановлением остатка, деление без восстановления остатка.
16. Алгоритм выполнения сложения в обратном и дополнительном кодах: сложение по модулю 2, с учетом знака и переполнения.
17. Современные процессоры выполняют операции вычитания в дополнительном (обратном) коде, чтобы обеспечить точность и скорость.
18. Основные логические операции с двоичными кодами: логическое суммирование, логическое умножение, логическое отрицание, суммирование по модулю два, логические сдвиги.
19. Логическое суммирование для чисел 110011100 и 10101011: 11101111.
20. Логическое умножение для чисел 110011100 и 10101011: 10001000.
21. Логическое отрицание для 110011100: 00110011.
22. Суммирование по модулю два для 110011100 и 10101011: 01100111.
23. Операция NAND для 110011100 и 10101011: 00011011.
24. Операция NOR для 110011100 и 10101011: 11001111.
25. Логический сдвиг отличается от арифметического сдвига тем, что не учитывает знак и переполнение.
26. Альтернативные названия логических операций: логическое суммирование (OR), логическое умножение (AND), логическое отрицание (NOT), суммирование по модулю два (XOR).
27. Методы вычисления сложных математических функций: аппроксимация, интерполяция, экстраполяция.
28. Методы вычисления квадратного корня: биномиальная теорема, метод Ньютона.
29. Простейший вариант вычисления тригонометрических функций: использование таблиц или приближения.
30. Ряды используются для вычисления математических функций, представляя их как бесконечную сумму.
31. Законы и правила алгебры Буля: закон коммутативности, закон ассоциативности, закон дистрибутивности.
32. Аксиомы булевой алгебры: закон тождества, закон противоречия, закон исключенного третьего.
33. Законы нулевого и универсального множества булевой алгебры: закон нулевого элемента, закон универсального элемента.
34. Теорема де Моргана: (A ∪ B)' = A' ∩ B'.
35. Законы повторения и двойной инверсии булевой алгебры: A ∪ A = A, A ∩ A = A, (A')' = A.
36. Коммуникативный и ассоциативный законы булевой алгебры: A ∪ B = B ∪ A, A ∩ B = B ∩ A, (A ∪ B) ∪ C = A ∪ (B ∪ C).
37. Дистрибутивный закон и закон поглощения булевой алгебры: A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C), A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C).
38. Правила положительных двоичных чисел, представленных в прямом коде: каждый разряд представляет степень 2.
39. Правила отрицательных двоичных чисел, представленных в прямом коде: каждый разряд представляет степень 2, знак определяется отдельно.
40. Правила положительных двоичных чисел, представленных в обратном коде: каждый разряд представляет степень 2, знак определяется отдельно.
41. Правила отрицательных двоичных чисел, представленных в обратном коде: каждый разряд представляет степень 2, знак определяется отдельно.
42. Правила положительных двоичных чисел, представленных в дополнительном коде: каждый разряд представляет степень 2, знак определяется отдельно.
43. Правила отрицательных двоичных чисел, представленных в дополнительном коде: каждый разряд представляет степень 2, знак определяется отдельно.
44. Логический сдвиг отличается от арифметического сдвига тем, что не учитывает знак и переполнение.
45. Арифметические сдвиги используются для выполнения арифметических операций, таких как умножение и деление.
46. Формат чисел с фиксированной точкой отличается от формата с плавающей точкой тем, что точка находится в фиксированном положении.
47. Арифметика для фиксированной или плавающей точки требует больше ресурсов от процессора, поскольку требует дополнительных операций для управления точкой.
48. Принцип двойственности заключается в том, что каждая логическая функция имеет двойственную функцию, полученную заменой операций и переменных.
49. Суперпозиция логических функций заключается в том, что каждая логическая функция может быть представлена как суперпозиция более простых функций.
50. Нормальная и совершенно нормальная форма логических функций: нормальная форма - функция, представленная в виде конъюнкции дизъюнкций, совершенно нормальная форма - функция, представленная в виде дизъюнкции конъюнкций.
51. Для логических функций получают совершенно нормальную форму логической функции, чтобы упростить ее представление и сделать более эффективным ее вычисление.
52. Количество вариантов нормальной и совершенно нормальной формы у любой логической функции равно 2^n, где n - количество переменных.
53. Алгоритм получения совершенно нормальной формы логической функции включает в себя замену операций и переменных, чтобы получить функцию в виде дизъюнкции конъюнкций.
54. Пример КНФ и СКНФ для трех переменных: КНФ - (A ∪ B) ∩ (A ∪ C), СКНФ - (A ∩ B) ∪ (A ∩ C).
55. Пример ДНФ и ДКНФ для трех переменных: ДНФ - A ∪ (B ∩ C), ДКНФ - (A ∪ B) ∩ (A ∪ C).
56. Двоично-десятичные числа хранятся в виде последовательности двоичных цифр, каждая из которых представляет одну десятичную цифру.
57. Двоично-десятичные числа имеют преимущества в том, что они могут быть легко преобразованы в двоичный формат, а недостатки в том, что они требуют больше памяти для хранения.
58. Алгоритм сложения чисел в двоично-десятичном формате включает в себя сложение соответствующих двоичных цифр и перенос при необходимости.
59. Алгоритм вычитания чисел в двоично-десятичном формате включает в себя вычитание соответствующих двоичных цифр и заем при необходимости.
60. Условия коррекции в двоично-десятичной арифметике включают в себя проверку на переполнение и коррекцию результата при необходимости.
61. Классификация систем счисления: позиционные и непозиционные системы счисления.
62. Позиционные системы счисления используются для представления чисел в виде последовательности цифр, каждая из которых представляет определенное значение.
63. Непозиционные системы счисления используются для представления чисел в виде комбинации цифр, каждая из которых представляет определенное значение.
64. Формула вычисления значения числа традиционной позиционной системы: значение = Σ(digit\_i \* base^i), где digit\_i - i-я цифра числа, base - основание системы счисления.
65. Способы перехода от одной системы позиционной исчисления к другой для целых чисел: метод деления, метод умножения.
66. Способы перехода от одной системы позиционной исчисления к другой для дробных чисел: метод деления, метод умножения.
67. Метод перехода с использованием особого соотношения оснований исходной и искомой систем счисления: метод использование коэффициента пересчета.
68. Переход от восьмеричной системы к шестнадцатеричной возможен с помощью метода умножения.
69. Особенность представления чисел в шестнадцатеричной системе исчисления заключается в том, что она использует буквы A-F для представления цифр 10-15.
70. Система исчисления, использующая буквы: шестнадцатеричная система исчисления.
71. Не позиционная система исчисления, которую мы использовали при решении задач: римская система счисления.
72. Суть метода деления (умножения) на новое основание заключается в том, что оно позволяет перейти от одной системы счисления к другой, используя деление или умножение на новое основание.
73. Кодирование алгебраических чисел для вычислительных систем: представление чисел в двоичном формате.
74. Способы представления отрицательных чисел в вычислительных системах: прямой код, обратный код, дополнительный код.
75. Дополнительный код отличается от обратного кода тем, что он использует дополнительное представление отрицательных чисел.
76. Обратный, дополнительный и прямой коды используются для представления отрицательных чисел в вычислительных системах.
77. При вычитании от меньшего числа большее получается дополнительный код.
78. Обратный, дополнительный и прямой коды имеют разный диапазон хранения данных при фиксированном размере места хранения.
79. Обратный и дополнительный коды имеют два представления нуля.
80. Представление числа -8 в дополнительном коде при однобайтном и двухбайтном представлении: 11111000 и 11111110 00000000.
81. Варианты перехода из дополнительного кода в прямой: метод инверсии и метод сложения.
82. Переход из обратного кода в прямой: метод инверсии.
83. Использование модифицированного кода в процессоре позволяет увеличить производительность путем оптимизации кода под конкретное аппаратное обеспечение.
84. Переполнение при выполнении арифметических операций определяется путем сравнения результата операции с диапазоном допустимых значений.
85. Переполнение – это ситуация, когда результат арифметической операции выходит за пределы допустимого диапазона значений. В микропроцессоре это определяется путем сравнения результата операции с маской переполнения.
86. Минимизация булевых функций проводится для упрощения схемы, уменьшения количества логических элементов и снижения задержек.
87. Основные понятия минимизации булевых функций включают в себя: алгебраическое выражение, Каноническое дизъюнктивное нормальное выражение (КДНВ), Каноническое конъюнктивное нормальное выражение (ККНВ), Карт Карно, Импликанта, Импликативная таблица, Импликативная нормальная форма (ИНФ), Дизъюнктивная нормальная форма (ДНФ), Конъюнктивная нормальная форма (КНФ), Минимизация по ДНФ, Минимизация по КНФ.
88. Наиболее известные методы минимизации булевых функций включают в себя: метод Карно, метод Квайна, метод Робинсона, метод Пэппи, метод Пурбаха.
89. Минимизация частично определенных булевых функций проводится для упрощения схемы, когда некоторые входные переменные не имеют значения.
90. Минимизация системы логических булевых функций проводится для упрощения схемы, когда несколько функций имеют общие входные переменные.
91. Алгоритм минимизации с использованием Карт Карно включает в себя: составление таблицы истинности, построение Карты Карно, объединение импликант, формирование результата минимизации.
92. Преимущества и недостатки минимизации с использованием Карт Карно включают в себя: простота и наглядность, ограничение на количество входных переменных, возможность ошибок при объединении импликант.
93. Правила объединения импликант в карте Карно включают в себя: объединение импликант с одинаковыми значениями, объединение импликант с противоположными значениями.
94. Правила построения карты Карно включают в себя: составление таблицы истинности, расстановка переменных в карте Карно, заполнение карты Карно значениями.
95. Какие значения могут быть в ячейках карты Карно включают в себя: 0, 1, X (неопределенное значение).
96. Правила формирования результата минимизации с картой Карно по «1» включают в себя: объединение импликант с значением 1, исключение импликант с значением 0.
97. Правила формирования результата минимизации с картой Карно по «0» включают в себя: объединение импликант с значением 0, исключение импликант с значением 1.
98. Булевы базис – это набор логических операций, на которых основываются все другие логические операции. Варианты включают в себя: базис И-НЕ, базис ИЛИ-НЕ.
99. Теоремой, руководствующейся при выборе булевого базиса, является теорема о полноте базиса.
100. Полная система логических функций (базис) включает в себя все возможные комбинации логических операций.
101. Преимущества базиса И-НЕ и ИЛИ-НЕ включают в себя: простота реализации, низкое потребление энергии.
102. Пример функции в базисе И-НЕ: Y = (A и B) или (C и не D).
103. Пример функции в базисе ИЛИ-НЕ: Y = (A или B) и (C или не D).
104. Основные понятия алгебры логики включают в себя: логическую переменную, логическую операцию, таблицу истинности, эквивалентность.
105. Способы задания логической функции включают в себя: таблицу истинности, дизъюнктивную нормальную форму (ДНФ), конъюнктивную нормальную форму (КНФ), каноническую дизъюнктивную нормальную форму (КДНФ), каноническую конъюнктивную нормальную форму (ККНФ).
106. Алгоритм минимизации Квайна включает в себя: составление таблицы истинности, составление импликантной таблицы, выбор импликант, формирование результата минимизации.
107. Импликантная таблица в методе Квайна нужна для определения минимального набора импликант, необходимых для реализации функции.
108. Составляется импликантная таблица в методе Квайна путем составления таблицы истинности и выделения импликант для каждого значения выходной переменной.
109. Тупиковая функция в методе Квайна – это функция, для которой нет импликант, соответствующих всем значениям выходной переменной.
110. Алгоритм выбора импликант из импликантной таблицы при минимизации методом Квайна включает в себя: выбор импликант с минимальным количеством единиц, исключение импликант, содержащих другие импликант.
111. Достоинства и недостатки метода Квайна включают в себя: простота и эффективность, ограничение на количество входных переменных, возможность ошибок при составлении импликантной таблицы.
112. Полная система логических функций включает в себя все возможные комбинации логических операций. Существует пять полных систем: базис И-НЕ, базис ИЛИ-НЕ, базис КОНЬЮНКЦИЯ-ИЛИ, базис КОНЬЮНКЦИЯ-ЭКВИВАЛЕНТНОСТЬ, базис ДИЗЬЮНКЦИЯ-ЭКВИВАЛЕНТНОСТЬ.
113. Последовательность в полной системе логических функций строится путем объединения базисных функций в более сложные функции.
114. Архитектура процессора – это совокупность компонентов, обеспечивающих выполнение операций над данными. Основные компоненты включают в себя: АЛУ, регистры, кэш, шина данных, шина управления, контроллер памяти, контроллер ввода-вывода.
115. Классификация процессоров включает в себя: по типу команд (RISC, CISC, MISC, VLIW), по типу архитектуры (суперскалярная, векторная, многопроцессорная), по типу памяти (с общим адресным пространством, с раздельными адресными пространствами), по типу тактирования (синхронные, асинхронные).
116. Набор команд – это набор инструкций, которые может выполнять процессор. Он определяет тип архитектуры процессора и влияет на его производительность.
117. Процессор в рамках одной архитектуры может отличаться по таким параметрам, как частота тактовой частоты, размер кэша, разрядность шины данных, тип памяти.
118. Шина процессора – это связь между процессором и другими устройствами в системе. Она определяет скорость передачи данных и количество устройств, которые могут быть подключены к процессору.
119. В многоуровневой архитектуре на аппаратном уровне определяется организация памяти, тип шины данных, тип кэша, тип регистров.
120. Классификация архитектур с точки зрения системы команд включает в себя: RISC, CISC, MISC, VLIW.
121. Классификация архитектур с точки зрения системы команд:
     1. CISC (Complex Instruction Set Computing): большое количество инструкций, которые могут выполняться за один такт, но занимают много места в памяти.
     2. RISC (Reduced Instruction Set Computing): небольшое количество инструкций, которые выполняются за несколько тактов, но занимают мало места в памяти.
     3. VLIW (Very Long Instruction Word): инструкции выполняются параллельно, что позволяет увеличить производительность.
     4. EPIC (Explicitly Parallel Instruction Computing): расширенная версия VLIW, которая поддерживает большее количество параллельных инструкций.
122. Отличительные признаки архитектур команд:
     1. CISC: большое количество инструкций, переменная длина инструкций, адресация по содержимому регистра, наличие микрокодов.
     2. RISC: небольшое количество инструкций, фиксированная длина инструкций, простая адресация, отсутствие микрокодов.
     3. VLIW: инструкции выполняются параллельно, длинные слова инструкций, отсутствие микрокодов.
     4. EPIC: расширенная версия VLIW, поддерживает большее количество параллельных инструкций.
123. Процессору с RISC архитектурой требуется меньше микрокоманд для вычисления простейшего арифметического выражения по сложению нескольких чисел, так как он имеет фиксированную длину инструкций и простую адресацию.
124. Процессоры с RISC архитектурой используются в основном в мобильных устройствах и серверах, где важна низкое энергопотребление и высокая производительность. Они не вытеснили процессоры CISC из-за их гибкости и возможности выполнять сложные инструкции.
125. Преимущества CISC процессоров: гибкость, возможность выполнять сложные инструкции, низкое энергопотребление. Недостатки: большие размеры микросхем, медленное выполнение инструкций.
126. CISC процессоры используются в основном в стационарных компьютерах и серверах, где важна гибкость и возможность выполнять сложные инструкции.
127. MISC (Minimal Instruction Set Computing) процессоры: небольшое количество инструкций, фиксированная длина инструкций, простая адресация, отсутствие микрокодов. Достоинства: низкое энергопотребление, высокая производительность. Где используются: в мобильных устройствах, встраиваемых системах.
128. VLIW процессоры: инструкции выполняются параллельно, длинные слова инструкций, отсутствие микрокодов. Достоинства: высокая производительность, низкое энергопотребление. Где используются: в суперкомпьютерах, графических процессорах.
129. Виртуальные архитектуры: абстракция над реальной аппаратной архитектурой, позволяющая программисту писать код, не учитывая особенности конкретной аппаратной платформы. Достоинства: простота программирования, портируемость кода. Недостатки: снижение производительности из-за абстракции.
130. Команды (инструкции) процессора: операции, которые могут быть выполнены процессором. Виды: арифметические, логические, перемещения данных, управления потоком выполнения.
131. Минимальный набор арифметических операций в АЛУ: сложение, вычитание, умножение, деление. Обычно АЛУ содержит больше операций для повышения производительности.
132. Инструкции процессору предоставляются программным обеспечением, а количество инструкций, которые могут быть выполнены за один такт, зависит от архитектуры процессора.
133. Тактирование процессора: управление временем выполнения инструкций с помощью тактового генератора. Влияет на производительность процессора. Сейчас используется для управления энергопотреблением и снижения шума.
134. Spectre и Meltdown: уязвимости в процессорах, позволяющие злоумышленникам получать доступ к конфиденциальным данным, хранящимся в памяти. Возникают из-за ошибок в дизайне процессора и неправильной реализации механизмов безопасности.
135. Сложность устранения уязвимостей типа Spectre и Meltdown заключается в том, что они требуют изменения дизайна процессора и могут повлиять на производительность.
136. Алгоритм сложения двух чисел в формате с плавающей точкой:
137. Выравнивание мантисс по порядку.
138. Сложение мантисс.
139. Нормализация результата.
140. Определение знака результата.
141. Алгоритм умножения двух чисел в формате с плавающей точкой:
142. Выравнивание мантисс по порядку.
143. Умножение мантисс.
144. Увеличение порядка результата на сумму порядков множителей.
145. Нормализация результата.
146. Определение знака результата.
147. Алгоритм деления двух чисел в формате с плавающей точкой:
148. Выравнивание мантисс по порядку.
149. Деление мантисс.
150. Уменьшение порядка результата на разность порядков делимого и делителя.
151. Нормализация результата.
152. Определение знака результата.
153. IEEE754: стандарт, определяющий формат представления чисел с плавающей точкой. Специальные числа: плюс/минус бесконечность, плюс/минус ноль, NaN (Not a Number).
154. Базовые форматы IEEE754: одинарная точность (32 бита), двойная точность (64 бита), расширенная двойная точность (80 битов). Форматы обмена: одинарная точность с обменом (32 бита), двойная точность с обменом (64 бита).
155. Расширенные и расширяемые форматы точности IEEE754: форматы с плавающей точкой, которые поддерживают большее количество битов для мантиссы и порядка.
156. Денормализованные числа IEEE754: числа с плавающей точкой, которые не могут быть представлены в нормализованном формате. Используются для представления очень малых чисел.
157. Подводные камни в арифметике с плавающей точкой: проблемы с точностью, округление, специальные числа, исключения.
158. Правила округления IEEE754: определение результата округления числа с плавающей точкой до заданной точности. Необходимые операции: округление, усечение, округление с направлением.
159. Обработка исключений IEEE754: механизм, позволяющий обрабатывать исключительные ситуации, возникающие при арифметических операциях с плавающей точкой, например, деление на ноль, переполнение и т.д.
160. Фрагменты формата с плавающей точкой: знак, порядок, мантисса.
161. Преимущества и недостатки формата с плавающей точкой: гибкость, широкий диапазон представления, проблемы с точностью, округление.
162. Стандарт IEEE 754 определяет операции сложения, вычитания, умножения и деления.
163. Необходимые операции, определяемые стандартом IEEE 754: сложение, вычитание, умножение и деление.
164. Есть две редакции стандарта IEEE 754: 1985 года и 2008 года. Основное различие в том, что редакция 2008 года добавляет поддержку новых форматов и операций.
165. Исключения стандарта IEEE 754: деление на ноль, переполнение, потеря точности, недопустимая операция, неверный формат.
166. Если при выполнении операций в формате с плавающей точкой АЛУ попадает на исключение, то оно генерирует сигнал исключения и записывает его в регистр состояния.
167. Два типа NaN в стандарте IEEE 754: qNaN (quiet NaN) и sNaN (signaling NaN). qNaN не генерирует сигнал исключения, а sNaN генерирует сигнал исключения.
168. Параллелизм нужен для увеличения производительности вычислений. Варианты построения параллелизма: параллельное выполнение инструкций, параллельное выполнение потоков, параллельное выполнение процессов.
169. Временный параллелизм позволяет выполнять несколько инструкций одновременно, что увеличивает производительность. Однако он ограничен ресурсами процессора.
170. Параллелизм реальных арифметических задач ограничен ресурсами процессора, памятью и скоростью передачи данных.
171. Матрица памяти - это устройство, которое хранит данные в виде матрицы. Она состоит из ячеек, каждая из которых может хранить одно значение.
172. Типы памяти: ОЗУ (оперативная память), ПЗУ (постоянная память), флеш-память, жесткий диск.
173. Запись-чтение из памяти происходит через шину, которая соединяет процессор с памятью.
174. Матрицы памяти могут быть классифицированы по типу доступа: последовательный доступ, произвольный доступ.
175. Основными классами ОЗУ являются SRAM (статическая оперативная память) и DRAM (динамическая оперативная память).
176. Энергозависимые запоминающие устройства требуют постоянного питания для сохранения данных.
177. Основные элементы ЭВМ: процессор, память, вход/выход.
178. АЛУ (арифметико-логическое устройство) - это часть процессора, которая выполняет арифметические и логические операции.
179. АЛУ выполняет арифметические и логические операции, а также управляет потоком данных.
180. Минимальный набор операций АЛУ: сложение, вычитание, умножение, деление.
181. Сумматор - это устройство, которое выполняет сложение двух чисел.
182. Бывают два типа сумматоров: полный сумматор и полусумматор.
183. Сумматоры могут быть реализованы с помощью логических элементов или арифметических схем.
184. Сумматоры с ускоренным переносом позволяют выполнять сложение более быстро.
185. Префиксный сумматор - это тип сумматора, который позволяет выполнять сложение более быстро.
186. Компаратор - это устройство, которое сравнивает два числа.
187. Компаратор равенства - это тип компаратора, который проверяет равенство двух чисел.
188. Компаратор больше/меньше - это тип компаратора, который проверяет, какое из двух чисел больше или меньше.
189. Языки описания аппаратуры: VHDL, Verilog.
190. Основные цели HDL: проектирование цифровых устройств, верификация цифровых устройств.
191. FPGA (полярно-программируемая логическая матрица) - это тип цифрового устройства, которое можно программировать.
192. FPGA позволяет выполнять цифровые операции более быстро, чем процессоры.
193. Современные вычислительные структуры с применением FPGA: параллельные вычисления, распределенные вычисления.
194. FPGA применяются в различных областях: телекоммуникации, авиация, автомобильная промышленность.
195. FPGA состоит из логических элементов, которые можно программировать.
196. Код Грея - это тип кода, который позволяет кодировать числа более эффективно.
197. Код Грея используется в различных областях: телекоммуникации, компьютерные сети.
198. Рефлексивный код Грея - это тип кода Грея, который позволяет кодировать числа более эффективно.
199. АЦП (аналогово-цифровой преобразователь) - это устройство, которое преобразует аналоговые сигналы в цифровые.
200. Основные параметры АЦП: разрешение, частота дискретизации, скорость преобразования.
201. Квантование - это процесс преобразования аналоговых сигналов в цифровые.
202. Дискретизация - это процесс преобразования аналоговых сигналов в цифровые.
203. Корректирующие коды - это тип кодов, которые позволяют исправлять ошибки при передаче данных.
204. Код Хэ́мминга - это тип корректирующего кода, который позволяет исправлять ошибки при передаче данных.
205. Контрольные суммы - это тип кодов, которые позволяют проверять целостность данных.
206. Система контроля - это тип системы, которая позволяет проверять целостность данных.
207. Систематические коды - это тип кодов, которые позволяют исправлять ошибки при передаче данных.
208. Условие обнаружение ошибки систематическим кодом - это тип условия, которое позволяет обнаруживать ошибки при передаче данных.
209. Простейший метод контроля ошибок - это тип метода, который позволяет проверять целостность данных.
210. Самоконтролирующиеся коды - это тип кодов, которые позволяют проверять целостность данных.
211. Код Хемминга - это тип самоконтролирующегося кода, который позволяет исправлять ошибки при передаче данных.
212. При использовании кода Хемминга, передается помимо информации контрольная сумма.
213. Количество контрольных бит кода Хемминга определяется длиной кода.
214. Логический вентиль - это тип логического элемента, который позволяет выполнять логические операции.
215. Электронный буфер - это тип логического элемента, который позволяет усиливать сигналы.
216. Логический вентиль AND - это тип логического элемента, который позволяет выполнять логическую операцию AND.
217. Логический вентиль OR - это тип логического элемента, который позволяет выполнять логическую операцию OR.
218. Исключающее ИЛИ - это тип логического элемента, который позволяет выполнять логическую операцию XOR.
219. Логический вентиль NAND - это тип логического элемента, который позволяет выполнять логическую операцию NAND.
220. Логический вентиль NOR - это тип логического элемента, который позволяет выполнять логическую операцию NOR.
221. Конъюнкция - это тип логической операции, которая позволяет выполнять логическую операцию AND.
222. Дизъюнкция - это тип логической операции, которая позволяет выполнять логическую операцию OR.
223. Исключающее ИЛИ - это тип логической операции, которая позволяет выполнять логическую операцию XOR.
224. Логический вентиль NAND - это тип логического элемента, который позволяет выполнять логическую операцию NAND.
225. Логический вентиль NOR - это тип логического элемента, который позволяет выполнять логическую операцию NOR.
226. Уровни питания цифровых микросхем: 5 В, 3,3 В, 1,8 В.
227. Логические уровни: высокий, низкий.
228. Логические уровни у приемника и источника могут быть разными.
229. Четыре основных семейства цифровых микросхем: TTL, CMOS, ECL, I2L.
230. Передаточная характеристика - это тип характеристики, которая описывает зависимость выходного сигнала от входного сигнала.
231. Статическая дисциплина - это тип дисциплины, которая описывает поведение цифровых схем в статическом режиме.
232. Комплементарная пара транзисторов - это тип схемы, которая позволяет усиливать сигналы.
233. Биполярные и КМОП транзисторы - это типы транзисторов, которые используются в цифровых схемах.
234. Идеальный и реальный источник тока - это типы источников тока, которые используются в цифровых схемах.
235. Конденсаторы - это типы элементов, которые используются в цифровых схемах для фильтрации сигналов.
236. Идеальный и реальный источник напряжения - это типы источников напряжения, которые используются в цифровых схемах.
237. Операционные усилители - это типы усилителей, которые используются в цифровых схемах для усиления сигналов.
238. Матрицы операционных усилителей - это типы схем, которые используются в цифровых схемах для усиления сигналов.
239. Основные операции, которые реализуют при помощи операционных усилителей: усиление, фильтрация, суммирование.
240. Вентиль НЕ - это тип логического элемента, который позволяет выполнять логическую операцию NOT.
241. Потребляемая мощность цифровых схем - это тип характеристики, которая описывает потребление энергии цифровыми схемами.
242. Псевдо n-МОП-логика - это тип логической схемы, которая позволяет выполнять логические операции.
243. Псевдо n-МОП-логика отличается от классического варианта тем, что она использует только n-МОП транзисторы.
244. Z состояние - это тип состояния, которое позволяет цифровым схемам работать в режиме ожидания.
245. Цифровые схемы можно подразделять на комбинационные и последовательные.
246. Комбинационные цифровые схемы - это типы схем, которые не имеют памяти.
247. Правила изображения цифровых схем: использование стандартных символов, указание входов и выходов.
248. Перемещать инверсию - это тип операции, которая позволяет упростить логические схемы.
249. Входы с Z состоянием - это тип входов, которые позволяют цифровым схемам работать в режиме ожидания.
250. Микросхема с тремя состояниями - это тип микросхемы, которая имеет три состояния: высокий, низкий, Z.
251. X - это тип символа, который используется для обозначения неизвестных значений.
252. При проектировании комбинационной логики можно использовать различные методы: метод Карно, метод Куайна-МакКласки.
253. Временные характеристики цифровых микросхем: задержка распространения, задержка реакции.
254. Временной график переключения цифровой микросхемы - это тип графика, который показывает зависимость выходного сигнала от входного сигнала.
255. Импульсные помехи - это тип помех, которые могут возникать при работе цифровых схем.
256. Задержка распространения - это тип характеристики, которая описывает время, необходимое для распространения сигнала через цифровую схему.
257. Задержка реакции - это тип характеристики, которая описывает время, необходимое для реакции цифровой схемы на входной сигнал.
258. Критический путь - это тип пути, который имеет максимальную задержку распространения.
259. Выбор варианта реализации цифровой схемы зависит от требований к схеме: скорость, потребляемая мощность, стоимость.
260. Базовые комбинационные блоки: мультиплексоры, демультиплексоры, дешифраторы.
261. Мультиплексоры и демультиплексоры используются для коммутации сигналов в цифровых системах.
262. Дешифратор - это устройство, которое преобразует двоичный код в десятичный. Обозначается как DEC.
263. Демультиплексор - это устройство, которое распределяет входной сигнал по нескольким выходам. Обозначается как DEMUX.
264. Дешифраторы используются в системах управления, где необходимо преобразовать двоичный код в десятичный.
265. Проектирование последовательной логики включает в себя разработку триггеров, регистров и других устройств, которые обрабатывают сигналы в последовательном режиме.
266. Типы триггеров: RS-триггер, D-триггер, JK-триггер.
267. D-триггер - это триггер, который имеет один вход и один выход. Преимущество синтеза схем на D-триггере заключается в том, что он позволяет создавать более сложные схемы с меньшим количеством элементов.
268. Последовательные схемы отличаются от комбинационных тем, что они обрабатывают сигналы в последовательном режиме, а не одновременно.
269. Временной график срабатывания по фронту и по спаду цифровой схемы показывает, когда сигнал меняет свое состояние.
270. RS-триггер построен на основе двух логических элементов "ИЛИ-НЕ".
271. RS-триггер имеет два состояния: 0 и 1.
272. Регистр - это устройство, которое хранит двоичный код. Бывают параллельные и последовательные регистры.
273. Защелка отличается от триггера тем, что она имеет только один вход.
274. Схема (цепь) определяется набором элементов и связей между ними. Параметры схемы характеризуют ее поведение.
275. Триггеры с функцией сброса используются для сброса триггера в начальное состояние. Обозначаются как CLR.
276. Триггеры с функцией разрешения используются для разрешения триггера на работу. Обозначаются как EN.
277. Синхронные схемы отличаются от асинхронных тем, что они обрабатывают сигналы в синхронном режиме, а не одновременно.
278. Синхронные последовательные схемы используются для обработки сигналов в последовательном режиме.
279. Конечный автомат - это устройство, которое может находиться в одном из нескольких состояний.
280. Алгоритм проектирования конечного автомата Мура включает в себя разработку таблицы состояний и таблицы переходов.
281. Алгоритм проектирования конечного автомата Мили включает в себя разработку таблицы состояний и таблицы выходов.
282. В узлах таблицы состояний указываются состояния, а около стрелок - переходы между состояниями.
283. При синтезе конечного автомата составляются таблицы состояний и таблицы переходов.
284. Автоматы Мура и Мили отличаются тем, что автомат Мура имеет выходные сигналы, зависящие от текущего состояния, а автомат Мили - от текущего состояния и входных сигналов.
285. Варианты кодирования состояний конечных автоматов: двоичное кодирование, десятичное кодирование.
286. Автоматы Мура дают выигрыш во времени, так как имеют выходные сигналы, зависящие от текущего состояния.
287. Декомпозиция конечных автоматов - это процесс разбиения автомата на более простые автоматы.
288. Декомпозиция конечных автоматов дает возможность упростить автомат и уменьшить количество элементов.
289. Алгоритм восстановления конечного автомата по электрической схеме включает в себя анализ схемы и разработку таблицы состояний.
290. Синхронизация последовательных схем необходима для обеспечения правильной работы схемы.
291. Временная характеристика работы цифровой системы - это график, показывающий, когда сигнал меняет свое состояние.
292. Расфазировка тактовых сигналов - это сдвиг фазы между тактовыми сигналами.
293. Для синхронизации современных микропроцессоров учитывается физическое расположение отдельных элементов.
294. Частота современных микропроцессоров стала расти медленно из-за физических ограничений.
295. Динамическая дисциплина - это набор правил, определяющих порядок обработки сигналов в цифровой системе.
296. Апертурное время последовательностных схем - это время, необходимое для обработки сигнала.
297. Период тактовых импульсов - это время, необходимое для обработки одного тактового импульса.
298. Расфазировка - это сдвиг фазы между тактовыми сигналами.
299. Метастабильность - это состояние, когда сигнал не может быть обработан из-за неопределенности его значения.
300. Регистр - это устройство, которое хранит двоичный код.
301. Бывают параллельные и последовательные регистры.
302. Регистры обозначаются на электрических схемах в разных стандартах.
303. Мультиплексоры необходимы для коммутации сигналов в цифровых системах.
304. Демультиплексоры необходимы для распределения входного сигнала по нескольким выходам.
305. Мультиплексоры обозначаются в разных стандартах.
306. Регистры процессора: регистры общего назначения, регистры индексные, регистры сдвига.
307. Регистры процессора необходимы для хранения двоичного кода.
308. Шины необходимы для передачи сигналов между элементами цифровой системы.
309. Бывают шины данных, шины адреса, шины управления.
310. Кэш - это устройство, которое хранит часто используемые данные.
311. Бывают кэши уровня 1, кэши уровня 2, кэши уровня 3.
312. Суперскалярная архитектура - это архитектура, которая позволяет выполнять несколько команд одновременно.
313. Особенность суперскалярной архитектуры - возможность выполнять несколько команд одновременно.
314. Предсказатели переходов необходимы для предсказания переходов между командами.
315. Иерархия памяти необходима для организации памяти в цифровой системе.
316. Гетерагенные вычисления - это вычисления, которые выполняются на разных типах процессоров.
317. FPGA-акселератор - это устройство, которое ускоряет выполнение команд на FPGA.
318. Сфера применения FPGA-акселераторов - ускорение выполнения команд в цифровых системах.
319. Особенность квантовых вычислений - возможность выполнять вычисления с использованием квантовых эффектов.
320. Квантовые вычисления могут помочь в решении задач, связанных с криптографией и оптимизацией.

Тензорные ядра - это устройства, которые выполняют вычисления с использованием тензоров.

1. Вычисления со смешанной точностью - это вычисления, которые выполняются с использованием разных типов данных.
2. Сфера применения тензорных ядер - ускорение выполнения команд в цифровых системах.
3. Тензорные вычисления эффективны при решении задач, связанных с машинным обучением и искусственным интеллектом.