Экзаменационный билет №1

Вопрос 1. Логарифмические частотные характеристики Цифровых Автоматических Систем (ЦАС): расчет, правила построения

Логарифмические частотные характеристики (ЛЧХ) цифровых автоматических систем — важный инструмент анализа динамических свойств систем управления, особенно при проектировании систем с обратной связью. Основная цель ЛЧХ — графическое представление амплитудно-частотной и фазо-частотной характеристик системы на логарифмической шкале частот, что удобно для охвата широкого диапазона частот и выявления особенностей поведения системы.

Расчет ЛЧХ цифровых систем начинается с получения передаточной функции системы в комплексной форме $W(z)$ или $W(s)$ (последнее применимо при аппроксимации непрерывным временем). В дискретных системах с периодом дискретизации T частота ω нормируется и переходит в комплексную плоскость $z = e^{j\omega T}$. Для каждого значения частоты ω рассчитывают амплитуду модуля передаточной функции $|W(e^{j\omega T})|$ и фазовый сдвиг $\arg W(e^{j\omega T})$.

Далее амплитудные значения переводят в логарифмическую шкалу, обычно в децибелах (дБ):

$$L(\omega) = 20\log_{10}|W(e^{j\omega T})|$$

Фазовый сдвиг выражается в градусах или радианах. Графики строятся на двух отдельных осях по частоте, отложенной в логарифмическом масштабе.

Правила построения ЛЧХ следующие:

1. **Выбор диапазона частот**: Обычно охватывается от низких частот (близких к нулю) до частот, близких к половине частоты дискретизации (частота Найквиста).

2. **Построение амплитудно-частотной характеристики (АЧХ)**: По оси X — логарифм частоты, по оси Y — амплитуда в дБ. Амплитуда обычно плавно меняется, с резкими переходами в точках полюсов и нулей.

3. **Построение фазо-частотной характеристики (ФЧХ)**: Фаза выводится на отдельном графике, выражается в градусах или радианах, показывает сдвиг выходного сигнала относительно входного.

4. **Использование правил асимптотического построения**: Для упрощения часто строят приближенные графики, используя асимптоты, основанные на частотах полюсов и нулей передаточной функции.

5. **Учет задержек и дискретизации**: В цифровых системах задержки и квантование могут влиять на форму ЛЧХ, что следует учитывать.

ЛЧХ позволяют оценивать устойчивость и качество управления, выявлять резонансные частоты, определять запас устойчивости по фазе и по амплитуде. Особенно важна для

проектирования регуляторов и корректоров, где наглядное понимание частотных свойств системы помогает выбрать параметры.

Вопрос 2. Проблема «исключающего ИЛИ» (XOR). Преодоление линейной разделимости

Проблема «исключающего ИЛИ» (XOR) — классическая задача в области машинного обучения и теории нейронных сетей, иллюстрирующая ограничения простейших моделей. Функция XOR возвращает 1, если входные биты различны, и 0, если одинаковы. В двумерном пространстве точек $\{(0,0), (0,1), (1,0), (1,1)\}$ результат нельзя разделить одной прямой, так как точки с разными классами чередуются диагонально.

Это иллюстрирует **проблему линейной разделимости** — простейшие модели, такие как перцептрон с одной линейной функцией активации, не могут корректно обучиться на XOR, так как их решения являются линейными границами.

Преодоление этой проблемы стало ключевым шагом в развитии нейронных сетей. Основные методы:

1. **Использование многослойных нейронных сетей (МНС)**. Добавление хотя бы одного скрытого слоя с нелинейной активацией позволяет моделировать нелинейные границы. Например, два нейрона первого слоя могут сформировать линейные границы, которые пересекаются, а выходной слой объединит их для реализации XOR.

2. **Использование нелинейных функций активации** (например, сигмоида, ReLU), что позволяет сети моделировать сложные разделяющие поверхности.

3. **Другие алгоритмы и методы**, включая ядерные методы в SVM, которые переводят данные в более высокоразмерное пространство, где задача становится линейно разделимой.

Таким образом, проблема XOR показала, что **простые линейные модели имеют ограниченный класс задач**, и стала стимулом для развития глубинного обучения и сложных архитектур.

Вопрос 3. Комбинационные и последовательностные логические устройства, их описание на языке Verilog

Комбинационные логические устройства

Комбинационные логические устройства — это такие схемы, в которых выходные сигналы зависят исключительно от текущих входных сигналов. Примеры включают логические элементы, мультиплексоры, дешифраторы и сумматоры.

В Verilog описания таких устройств часто используют оператор `assign` для создания непрерывных связей. Например, логический элемент AND:

```
module and_gate(
    input wire a,
```

```
    input wire b,
    output wire y
);
    assign y = a & b; // Выход равен логическому И входов
endmodule
```

Для более сложных устройств, таких как 4-битный сумматор, можно использовать битовые операции:

```
module adder_4bit(
    input wire [3:0] a,
    input wire [3:0] b,
    output wire [3:0] sum,
    output wire carry_out
);
    assign {carry_out, sum} = a + b; // Сложение с учётом переноса
endmodule
```

Экзаменационный билет №2

Вопрос 1. Дискретные передаточные функции элементов цифровых автоматических систем (ЦАС). Экстраполяторы и их передаточные функции

Дискретные передаточные функции являются математическим описанием динамики элементов цифровых автоматических систем (ЦАС) в z-преобразовании. Они отражают связь между входом и выходом элемента в дискретном времени с учетом периода дискретизации $ T $. Передаточная функция в области z — это отношение z-преобразования выходного сигнала к входному при нулевых начальных условиях.

Общие дискретные передаточные функции часто записываются в виде дробно-рациональных выражений:

$$W(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}}$$

где коэффициенты $ b_i $, $ a_j $ характеризуют свойства элемента.

Основные элементы ЦАС включают:

- **Звено первого порядка** — дискретный аналог непрерывного первого порядка, его передаточная функция может иметь вид:

$$W(z) = \frac{K(1-\alpha)}{1 - \alpha z^{-1}}$$

где $ = e^{-T/} $, $ $ — постоянная времени.

- **Интегрирующее звено**, которое накапливает входной сигнал, имеет передаточную функцию:

$$W(z) = \frac{KT}{1 - z^{-1}}$$

- **Дифференцирующее звено** можно получить из разности последовательных отсчетов.

Экстраполяторы — это специальные дискретные устройства, которые используют известные значения сигнала для прогнозирования его будущих значений. В ЦАС экстраполяторы применяются для повышения точности и компенсации задержек.

Простейшие экстраполяторы можно представить как конечные разностные уравнения, аппроксимирующие производные или интегралы функции. Их передаточные функции в z-представлении могут иметь вид:

$$W(z) = \frac{1}{1 - z^{-1}} \quad \text{(интегратор)}$$

или более сложные, например, линейные экстраполяторы первого порядка:

$$W(z) = \frac{1 - \beta z^{-1}}{1 - \alpha z^{-1}}$$

где α, β — параметры, определяющие свойства фильтра.

Экстраполяторы широко используются в цифровой обработке сигналов для сглаживания, предсказания и фильтрации.

Вопрос 2. Цель обучения нейронных сетей. Обучение с учителем. Обучение без учителя

Цель обучения нейронных сетей — автоматическое нахождение таких параметров (весов и смещений), которые обеспечивают минимизацию ошибки между желаемым (эталонным) откликом и выходом сети. Другими словами, обучение направлено на построение модели, которая адекватно аппроксимирует или классифицирует входные данные.

Существуют два основных типа обучения:

- **Обучение с учителем (supervised learning)** — модель обучается на размеченных данных, где каждому входу соответствует известный правильный выход. Во время обучения нейросеть получает входные данные и эталонные ответы, и корректирует свои веса, чтобы минимизировать ошибку (например, используя метод градиентного спуска и алгоритм обратного распространения ошибки). Это позволяет сети впоследствии делать точные прогнозы или классификацию на новых данных.

- **Обучение без учителя (unsupervised learning)** — в этом случае данные не содержат меток. Цель обучения — выявление скрытых структур, закономерностей или группировок в данных. Популярные методы — кластеризация, самоорганизующиеся карты (например, карты Кохонена). Обучение происходит путем адаптации весов с учетом сходства или различий между входами, без прямой целевой функции.

Кроме того, существуют гибридные подходы и обучение с подкреплением.

Обучение нейросетей позволяет решать широкий спектр задач: распознавание образов, прогнозирование, управление и многое другое.

Вопрос 3. Маршрут проектирования в САПР Quartus II

САПР Quartus II компании Intel (ранее Altera) предназначена для проектирования цифровых схем и программирования FPGA и CPLD. Основной маршрут проектирования включает несколько ключевых этапов, от создания проекта до загрузки конфигурации в устройство.

Основные этапы проектирования

1. **Создание нового проекта**
    - В Quartus II откройте “New Project Wizard”.

- Укажите директорию проекта, имя проекта и основное устройство FPGA/CPLD.
- Добавьте исходные файлы, если они уже созданы.

2. **Добавление исходного кода**
   - Напишите HDL-код (Verilog или VHDL) для описания логики устройства.
   - Для схемотехнического подхода используйте встроенный инструмент Block Diagram/Schematic Editor.
3. **Назначение пинов устройства**
   - Используйте Pin Planner для привязки логических сигналов к физическим выводам FPGA.
   - Задайте электрические характеристики пинов (например, стандарт сигналов, уровень напряжения).
4. **Синтез проекта**
   - Выполните компиляцию проекта (synthesis), чтобы преобразовать HDL-код в логические элементы устройства.
   - Проверьте сообщения о предупреждениях и ошибках.
5. **Анализ временных характеристик**
   - После компиляции используйте TimeQuest Timing Analyzer для проверки временных ограничений.
   - Убедитесь, что проект соответствует заданным требованиям (например, максимальная частота).
6. **Симуляция проекта**
   - Для функциональной проверки используйте встроенный инструмент ModelSim или сторонний симулятор.
   - Проверьте корректность работы схемы на основе тестовых входных данных.
7. **Компиляция и оптимизация**
   - После внесения изменений в проект выполните финальную компиляцию.
   - Используйте отчёты Quartus II для анализа использования ресурсов и возможной оптимизации логики.
8. **Создание конфигурационного файла**
   - Сгенерируйте файл конфигурации (например, .sof или .pof) для загрузки в целевое устройство.
9. **Программирование FPGA**
   - Используйте Programmer для загрузки конфигурации в FPGA.
   - Подключите устройство через JTAG, USB-Blaster или другой интерфейс и выполните программирование.
10. **Тестирование и отладка**
    - Проверьте работу схемы на аппаратном уровне.
    - Используйте SignalTap Logic Analyzer для анализа внутренних сигналов FPGA в реальном времени.

Особенности маршрута проектирования в Quartus II

- **Поддержка IP-ядер**: возможность добавления готовых функциональных модулей, таких как процессоры Nios II, контроллеры памяти и интерфейсы.
- **Удобная интеграция**: инструменты для работы с внешними симуляторами и средствами отладки.
- **Многоуровневая оптимизация**: настройки для достижения наилучшей производительности или минимального использования ресурсов.

Маршрут проектирования в Quartus II предоставляет гибкие возможности для создания цифровых систем любой сложности, от простых логических схем до высокопроизводительных встроенных решений.

Экзаменационный билет №3

Вопрос 1. Частотные методы исследования цифровых автоматических систем (ЦАС). Реакция элемента ЦАС на гармоническое входное воздействие

Частотные методы исследования ЦАС базируются на анализе поведения системы при входных сигналах гармонической формы. Этот подход позволяет изучать устойчивость, динамику и качество регулирования системы.

В цифровых автоматических системах, работающих с дискретными сигналами, частотный анализ обычно проводится с использованием z-преобразования, что позволяет перейти в частотную область, аналогично непрерывному преобразованию Лапласа.

Основная идея — исследовать реакцию элемента ЦАС на входной сигнал вида:

$$x(k) = A \cdot \sin(\omega k T + \phi)$$

где $ $ — частота в радианах, $ T $ — период дискретизации.

Реакция системы на гармонический сигнал представляется выходным сигналом той же частоты, но с измененной амплитудой и сдвигом фазы. Для линейных дискретных систем эта реакция полностью описывается передаточной функцией $ W(z) $ на единичной окружности $ z = e^{jT} $.

Амплитудно-частотная характеристика (АЧХ) — это модуль значения $ W(e^{jT}) $, показывающий, насколько изменяется амплитуда сигнала на выходе относительно входа при заданной частоте.

Фазо-частотная характеристика (ФЧХ) — аргумент $ W(e^{jT}) $, показывающий фазовый сдвиг выходного сигнала относительно входного.

Частотный анализ позволяет выявить резонансы, устойчивость и поведение системы в широком диапазоне частот.

Вопрос 2. Алгоритм обучения однослойного персептрона (метод Розенблатта)

Однослойный персептрон — простейшая модель искусственного нейрона, которая классифицирует входные данные на два класса, используя линейное разделение.

Метод Розенблатта — классический алгоритм обучения персептрона с учителем. Его цель — подобрать веса $ $ и порог $ $, чтобы сеть правильно классифицировала обучающую выборку.

Алгоритм выполняется итеративно:

1. Инициализация весов случайным образом или нулями.

2. Для каждого обучающего примера $ (, d) $, где $ $ — вход, $ d {+1, -1} $ — желаемый выход:
    - Вычисляется выход персептрона:

$$y = \text{sign}(\mathbf{w} \cdot \mathbf{x} - \theta)$$

3. Если $ y d $, веса корректируются по правилу:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta(d - y)\mathbf{x}$$

где $ $ — коэффициент обучения.

4. Процесс повторяется до тех пор, пока все примеры не классифицируются правильно или не достигнут лимит итераций.

Алгоритм Розенблатта гарантирует сходимость, если данные линейно разделимы. В противном случае он не сможет найти решение.

Вопрос 3. Язык Verilog. Структура описания проекта на Verilog

Язык Verilog — это язык описания аппаратуры (Hardware Description Language, HDL), предназначенный для моделирования и проектирования цифровых систем. Основная цель Verilog — описывать поведение цифровой схемы и её структуру на высоком уровне абстракции.

Структура проекта на Verilog

Проект на Verilog обычно включает несколько модулей (“modules”), которые являются основными строительными блоками описания. Модуль может включать:

1. **Заголовок модуля**
    - Определение имени модуля и его интерфейса (порты ввода/вывода):

```
module module_name(input wire in1, output wire out1);
```

2. **Объявление сигналов и переменных**
    - Внутренние сигналы, регистры и константы, необходимые для реализации логики:

```
wire internal_signal;
reg [3:0] counter;
```

3. **Поведенческое или структурное описание**
    - Поведенческое описание использует операторы `always`, `initial` или `assign` для определения логики. Пример:

```
always @(posedge clk) begin
    if (reset)
        counter <= 0;
    else
```

```
        counter <= counter + 1;
    end
```

- Структурное описание соединяет различные модули:

```
module top_module(input wire clk, reset, output wire [3:0] out);
    wire internal_signal;
    submodule u1 (.clk(clk), .reset(reset), .out(internal_signal));
endmodule
```

4. **Конец модуля**
    - Завершается ключевым словом `endmodule`.

Проект может включать дополнительные файлы, содержащие тестовые окружения (testbenches) для проверки работы модулей.

Экзаменационный билет №4

Вопрос 1. Передаточная функция приведенной непрерывной части системы управления

Передаточная функция — это фундаментальное понятие теории управления, позволяющее описывать поведение системы в частотной области. Для непрерывной системы управления она определяется как отношение выходного сигнала к входному в изображенной области Лапласа при нулевых начальных условиях. В инженерной практике часто применяется понятие *приведенной* передаточной функции, особенно при анализе смешанных систем, содержащих как дискретные, так и непрерывные элементы.

Под приведенной передаточной функцией непрерывной части системы управления понимается передаточная функция, отражающая только динамику непрерывных звеньев, исключая влияние дискретных компонентов и других подсистем, которые можно рассматривать отдельно. Обычно это нужно для раздельного анализа частей сложной системы и последующего синтеза управления.

Рассмотрим пример. Пусть система включает несколько последовательных звеньев, таких как интегратор, усилитель, фильтр. Каждое звено имеет свою передаточную функцию, например:

$$W_1(s) = \frac{K}{s}, \quad W_2(s) = \frac{1}{Ts+1}, \quad W_3(s) = K_f$$

Приведенная передаточная функция — это их произведение:

$$W(s) = W_1(s) \cdot W_2(s) \cdot W_3(s)$$

Она показывает, как выход системы зависит от входа, если игнорировать влияние дискретных или внешних компонентов, например, блока дискретного управления.

Приведенная передаточная функция позволяет: - Анализировать устойчивость непрерывной части системы. - Определять переходные характеристики. - Выполнять синтез корректирующих устройств для обеспечения требуемых свойств.

Таким образом, передаточная функция приведенной непрерывной части является важным инструментом для упрощенного анализа и проектирования систем управления.

Вопрос 2. Процедура обратного распространения ошибки. Сходимость алгоритма обратного распространения ошибки и способы ее ускорения

Обратное распространение ошибки (*backpropagation*) — базовый алгоритм обучения многослойных нейронных сетей. Его цель — минимизация функции ошибки (например, среднеквадратичной ошибки между выходом сети и эталонным значением) за счет корректировки весов нейронов.

Алгоритм включает следующие этапы:

1. **Прямой проход**: вычисляются выходы всех нейронов, начиная от входного слоя до выходного.
2. **Вычисление ошибки на выходном слое**: разность между целевым значением и фактическим выходом.
3. **Обратный проход (обратное распространение)**: ошибка передается назад по слоям, а градиенты функции ошибки вычисляются по правилу дифференцирования сложных функций (цепное правило).
4. **Обновление весов**: веса корректируются в направлении, противоположном градиенту, с использованием выбранного коэффициента обучения $ $:

$$w_{ij}^{(new)} = w_{ij}^{(old)} - \eta \frac{\partial E}{\partial w_{ij}}$$

где $ E $ — функция ошибки.

Сходимость алгоритма обратного распространения ошибки

Алгоритм гарантирует сходимость при условии, что: - Функция ошибки непрерывна и дифференцируема. - Коэффициент обучения $ $ достаточно мал.

Однако на практике возможны проблемы: - **Медленная сходимость**: градиент может быть очень мал, особенно в глубоких сетях (проблема затухающего градиента). - **Застревание в локальных минимумах**.

Способы ускорения сходимости

1. **Инициализация весов**: веса должны быть малыми случайными значениями (например, по методу Хе или Глорот).
2. **Импульс (momentum)**: добавляется член, учитывающий предыдущее изменение весов, что помогает преодолевать плато и локальные минимумы:

$$\Delta w_{ij}(t) = -\eta \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(t-1)$$

где $ $ — коэффициент импульса.

3. **Адаптивные методы обучения**: использование алгоритмов, таких как RMSProp, Adam, AdaGrad, которые адаптируют скорость обучения для каждого параметра.
4. **Нормализация входов**: помогает избежать слишком больших или малых градиентов.
5. **Использование мини-батчей**: обновление градиента на части обучающей выборки ускоряет процесс.

Таким образом, процедура обратного распространения ошибки является основным методом обучения нейронных сетей, а ускорение ее сходимости достигается за счет комбинации методов оптимизации, правильной инициализации и настройки гиперпараметров.

Вопрос 3. Язык Verilog. Выражения. Операторы

Выражения

Выражения в Verilog описывают комбинационную логику или вычисления. Они состоят из операндов и операторов.

Операторы в Verilog:

1. **Арифметические**: +, -, *, /, %.
2. **Логические**: &&, ||, !.
3. **Побитовые**: &, |, ^, ~.
4. **Сравнения**: ==, !=, >, <, >=, <=.
5. **Шифты**: <<, >>.
6. **Тернарный оператор**:
    - `? :` для условных выражений: `out = (a > b) ? a : b;`
7. **Присваивание**:
    - `=` (блоки `initial`), <= (неблокирующее присваивание в `always`).

Пример:

```
assign sum = a + b;
assign equal = (a == b) ? 1'b1 : 1'b0;
```

Экзаменационный билет №5

Вопрос 1. Построение переходных процессов. Использование общей формулы обратного Z-преобразования

Переходный процесс в цифровой системе автоматического управления (ЦАС) — это реакция системы на начальные условия или на скачкообразное изменение входного сигнала. Анализ переходных процессов позволяет оценить такие характеристики системы, как устойчивость, быстродействие, перерегулирование и колебательность.

Основой построения переходной характеристики в цифровой системе является использование Z-преобразования. После получения передаточной функции системы в z-области можно определить отклик системы на заданное входное воздействие, например, единичный скачок (ступенчатое воздействие), и перейти к временной области с помощью **обратного Z-преобразования**.

Общая формула обратного Z-преобразования:

$$x(k) = \frac{1}{2\pi j} \oint_C X(z) z^{k-1} dz$$

где контур интегрирования C — замкнутая кривая, охватывающая все полюса функции $ X(z) $ и лежащая внутри области сходимости.

Однако в инженерной практике обратное Z-преобразование чаще выполняется: - По таблицам стандартных Z-преобразований. - С использованием разложения функции на простейшие дроби. - Через метод вычитов, если используется ручной расчет. - С помощью программных средств (например, MATLAB), где используется функция `iztrans`.

Построение переходной характеристики выполняется следующим образом: 1. Получить передаточную функцию $ W(z) = $. 2. Задать входное воздействие, например, $ U(z) = $ для единичного скачка. 3. Найти выход $ Y(z) = W(z) U(z) $. 4. Выполнить обратное Z-преобразование $ y(k) = Z^{-1}{Y(z)} $, получив выражение во временной области. 5. Построить график зависимости $ y(k) $ от дискретного времени $ k $.

Таким образом, применение обратного Z-преобразования позволяет получить переходную характеристику и анализировать поведение цифровой системы во времени.

Вопрос 2. Однослойные и многослойные нейронные сети

Нейронная сеть — это модель, вдохновлённая биологическим мозгом, состоящая из множества искусственных нейронов, соединённых между собой весами. Основное отличие между однослойными и многослойными сетями заключается в их способности решать линейно или нелинейно разделимые задачи.

Однослойные нейронные сети

Однослойная нейронная сеть (персептрон) состоит из одного слоя вычислительных нейронов, напрямую соединённых с входами. Такой персептрон способен решать

задачи линейной классификации, то есть разделять выборки с помощью гиперплоскости. Веса настраиваются с помощью алгоритма обучения, например, метода Розенблатта.

Ограничения: - Не способен решать задачи, где классы не разделяются прямой (например, задача XOR). - Простая архитектура, быстрая сходимость.

Многослойные нейронные сети

Многослойная нейронная сеть (MLP — многослойный персептрон) включает один или несколько скрытых слоёв между входами и выходом. Каждый нейрон скрытого слоя получает сигналы от всех нейронов предыдущего слоя. За счёт использования **нелинейных функций активации** (например, ReLU, сигмоида, tanh) многослойные сети способны аппроксимировать любые непрерывные функции и решать задачи с **нелинейной разделимостью**.

Обучение многослойных сетей осуществляется методом обратного распространения ошибки (backpropagation) с применением градиентного спуска.

Преимущества: - Гибкость и высокая аппроксимирующая способность. - Возможность обработки сложных зависимостей во входных данных. - Применимы к задачам классификации, регрессии, распознавания образов и др.

Недостатки: - Увеличение вычислительных затрат при росте числа слоёв. - Риск переобучения при недостатке данных. - Необходимость подбора гиперпараметров (число слоёв, нейронов, скорость обучения и др.).

Таким образом, однослойные сети эффективны для простых задач, но для более сложных случаев необходима глубокая архитектура, обеспечивающая большую выразительную силу модели.

Вопрос 3. Язык Verilog. Подпрограммы

Подпрограммы позволяют переиспользовать код и упрощают проектирование.

Виды подпрограмм:

1. **Функции (`function`)**:
    - Возвращают одно значение.
    - Используются для выполнения небольших вычислений.

```
function [3:0] add;
    input [3:0] a, b;
    begin
        add = a + b;
    end
endfunction
```

2. **Задачи (`task`)**:
    - Могут возвращать несколько значений через выходные параметры.

- Поддерживают задержки (#, @):

```
task delay_task;
    input [7:0] value;
    begin
        #10;
        $display("Value: %d", value);
    end
endtask
```

Экзаменационный билет №6

Вопрос 1. Построение переходных процессов. Расчет на основе разложения передаточной функции САУ на элементарные дроби

Переходный процесс в системе автоматического управления (САУ) характеризует поведение системы во времени после приложения входного воздействия, например, ступенчатого сигнала. Для анализа переходных процессов в дискретных системах часто используется метод разложения передаточной функции на элементарные дроби.

Передаточная функция системы в z-области имеет вид:

$$W(z) = \frac{Y(z)}{U(z)} = \frac{N(z)}{D(z)}$$

где $N(z)$ и $D(z)$ — многочлены. Чтобы получить отклик системы во временной области, необходимо выполнить обратное Z-преобразование. Один из способов — разложить дробь на сумму простейших дробей:

$$W(z) = \sum \frac{A}{z - p}$$

где p — полюса системы, A — соответствующие коэффициенты (выResidue вычисляются с помощью метода неопределённых коэффициентов или вычетов). После разложения каждое слагаемое соответствует известной форме обратного Z-преобразования из таблиц.

Процедура построения переходного процесса:

1. Разложить $W(z)$ на элементарные дроби.
2. Для каждого слагаемого выполнить обратное Z-преобразование, используя таблицы.
3. Найти отклик на заданное входное воздействие, например, единичный скачок.
4. Построить график отклика системы.

Метод удобен, так как позволяет аналитически находить временные зависимости для каждого слагаемого, а значит, для всей системы.

Вопрос 2. Персептрон (решаемые задачи, представимость, линейная разделимость)

Персептрон — это простейшая модель нейронной сети, состоящая из одного слоя вычислительных элементов, соединённых с входными сигналами. Каждый входной сигнал умножается на вес, затем вычисляется взвешенная сумма, и результат передаётся через функцию активации (обычно пороговую).

Персептрон решает задачи **линейной классификации**, то есть задачи, где классы можно разделить прямой (в 2D) или гиперплоскостью (в n-мерном пространстве). Примеры таких задач:

- Определение, принадлежит ли точка области (линейная граница).
- Классификация объектов на два класса по линейным признакам.

Однако персептрон **не способен** решать задачи, которые не линейно разделимы, например задачу XOR.

Теорема о представимости (Cover's Theorem) утверждает, что для линейно разделимой задачи существует множество весов, которые позволяют правильно классифицировать обучающую выборку. В случае нелинейной разделимости однослойный персептрон бесполезен.

Таким образом, персептрон применим для простых задач, где достаточно линейной границы между классами. Для более сложных задач требуется многослойная нейросеть.

Вопрос 3. Аппаратные ядра ARM Cortex Cortex-A9 и Cortex-A53

Cortex-A9

ARM Cortex-A9 — это 32-битное ядро архитектуры ARMv7-A, ориентированное на системы с высокой производительностью. Основные характеристики:

- Суперскалярная архитектура с двумя или более исполнительными конвейерами.
- Поддержка SIMD-инструкций (NEON).
- Поддержка аппаратного разделения памяти и многозадачности.
- Предназначено для встраиваемых решений, таких как мультимедийные устройства, сетевое оборудование, одноплатные компьютеры.

Cortex-A9 применяется в системах, где требуется высокая производительность при умеренном энергопотреблении. Примеры использования: процессоры NVIDIA Tegra, Texas Instruments OMAP.

Cortex-A53

ARM Cortex-A53 — 64-битное ядро архитектуры ARMv8-A. Отличается следующими особенностями:

- Поддержка 64-битных вычислений и инструкций.
- Архитектура с улучшенной энергоэффективностью.
- Поддержка виртуализации, аппаратного шифрования, улучшенного управления памятью.
- Применяется в мобильных устройствах, одноплатных компьютерах (например, Raspberry Pi 3), а также в некоторых серверах.

Cortex-A53 обычно используется в энергосберегающих системах, где важен баланс между производительностью и энергопотреблением. В конфигурациях big.LITTLE может

сочетаться с более мощными ядрами, такими как Cortex-A72, обеспечивая оптимальную производительность в зависимости от нагрузки.

Оба ядра (A9 и A53) предназначены для встраиваемых и мобильных систем, но Cortex-A53 представляет собой более современное и энергоэффективное решение с поддержкой 64-битных вычислений.

Экзаменационный билет №7

Вопрос 1. Расчет аналогового прототипа корректирующего устройства при наличии неизменяемой непрерывной части системы методом логарифмических частотных характеристик

При проектировании корректирующего устройства (КУ) для системы автоматического управления (САУ) часто используется метод логарифмических амплитудно-частотных характеристик (ЛАЧХ), который позволяет наглядно определить влияние отдельных звеньев на поведение системы. Особое внимание уделяется случаю, когда в системе присутствует неизменяемая непрерывная часть — например, физический процесс, исполнительный механизм или датчик, параметры которых заданы и не подлежат изменению.

Процедура расчета включает несколько этапов:

1. **Построение ЛАЧХ неизменяемой части** — определяется амплитудно-частотная характеристика данной части системы. Это делается либо экспериментально, либо по известной передаточной функции.

2. **Формулировка требований к системе** — определяются показатели качества, такие как запас устойчивости по амплитуде и фазе, требуемая полоса пропускания, форма переходного процесса.

3. **Определение формы ЛАЧХ корректирующего устройства** — производится анализ требуемой ЛАЧХ замкнутой системы. Обычно это графическое наложение желаемой ЛАЧХ на ЛАЧХ неизменяемой части, чтобы понять, какую характеристику должен иметь КУ для выполнения требований.

4. **Синтез корректирующего устройства** — по разности между требуемой ЛАЧХ и ЛАЧХ неизменяемой части определяется передаточная функция КУ. Формируются типовые звенья: интеграторы, дифференциаторы, фильтры и усилители. Этот прототип описывается в аналоговой форме, и его можно реализовать с помощью операционных усилителей или других аналоговых компонентов.

5. **Проверка полученной схемы** — проводится моделирование или экспериментальная проверка, чтобы убедиться, что корректирующее устройство в сочетании с неизменяемой частью системы дает требуемый результат по частотным и переходным характеристикам.

Метод ЛАЧХ позволяет эффективно учитывать ограничения и особенности системы, а также упрощает синтез КУ, делая его более наглядным и интуитивным.

Вопрос 2. Обучение когнитрона

Когнитрон — это тип многослойной нейронной сети, разработанный Кунио Фукусимой в 1975 году для распознавания образов. Основная идея когнитрона — это моделирование

процессов обработки информации в зрительной коре человека, а обучение сети происходит без явного учителя, с использованием принципов самоорганизации.

Архитектура когнитрона:

- **Элементарные слои**: нижние уровни сети выделяют простые признаки изображения (линии, углы, кривизны).
- **Объединяющие слои**: более высокие уровни комбинируют простые признаки в более сложные образы.
- **Слои распознавания**: заключительные уровни принимают решения о классификации образов.

Принцип обучения:

- **Без учителя**: обучение происходит на основе самоорганизации весов, когда нейроны усиливают реакции на часто встречающиеся паттерны.
- **Использование латерального подавления**: активные нейроны подавляют соседние, чтобы выделить наиболее выраженные признаки.
- **Фиксация связей**: после нескольких итераций обучения веса стабилизируются, формируя «навыки» распознавания.

Когнитрон способен самостоятельно обучаться выделению признаков и классификации, что делает его мощным инструментом для задач компьютерного зрения.

Вопрос 3. ПЛИС типа FPGA фирмы Altera семейств Cyclone, Arria и Stratix

FPGA (Field-Programmable Gate Array) фирмы Altera (ныне Intel) представляют собой программируемые логические интегральные схемы, которые позволяют реализовывать цифровые схемы с высокой степенью гибкости. Каждое семейство FPGA Altera ориентировано на разные сегменты задач:

Cyclone

- **Назначение**: массовое производство, недорогие приложения.
- **Особенности**: низкое энергопотребление, базовый набор логических элементов, поддержка PLL, встроенные блоки памяти и мультипликаторы.
- **Применение**: бытовая электроника, недорогие системы управления, обработка сигналов с невысокими требованиями к производительности.

Arria

- **Назначение**: средний уровень сложности задач.
- **Особенности**: улучшенная производительность, поддержка высокоскоростных интерфейсов (Transceivers), большая плотность логических элементов.
- **Применение**: системы обработки сигналов, коммуникационное оборудование, промышленные системы.

Stratix

- **Назначение**: высокопроизводительные системы.
- **Особенности**: максимальная плотность логических элементов, большое количество встроенных DSP-блоков, поддержка многогигабитных интерфейсов, высокая энергоэффективность для мощных приложений.
- **Применение**: телекоммуникации, обработка видео, большие системы обработки данных и вычислений.

Таким образом, выбор семейства FPGA зависит от сложности и задач проекта: Cyclone — для простых решений, Arria — для среднего уровня, Stratix — для максимально производительных систем.

Экзаменационный билет №8

Вопрос 1. Устойчивость ЦАС. Критерий устойчивости Найквиста

Устойчивость системы автоматического управления (ЦАС) — одно из основных требований при проектировании систем. Система считается устойчивой, если при любом ограниченном входном воздействии её выходное воздействие остаётся ограниченным, и после окончания воздействия система возвращается в равновесное состояние.

Для анализа устойчивости широко используется частотный метод, в частности, критерий Найквиста. Он основан на рассмотрении комплексной частотной характеристики разомкнутой системы (передаточной функции разомкнутого контура) и её отображении на комплексной плоскости.

$$F(\mathrm{j}\omega) = 1 + W(\mathrm{j}\omega) = \frac{L(\mathrm{j}\omega) + N(\mathrm{j}\omega)}{L(\mathrm{j}\omega)}$$

Где $W(\mathrm{j}\omega) = \frac{N(\mathrm{j}\omega)}{L(\mathrm{j}\omega)}$ — частотная передаточная функция разомкнутой системы

Система устойчива при

$$\Delta \arg \mathrm{F}(\mathrm{j}\omega) = \Delta \arg [\mathrm{L}(\mathrm{j}\omega) + \mathrm{N}(\mathrm{j}\omega)] - \Delta \arg \mathrm{L}(\mathrm{j}\omega) = 0 \text{ при } 0 \le \omega \le +\infty$$

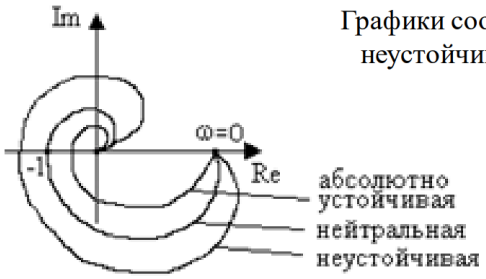
Об изменении аргумента вектора F(jω) удобнее судить по годографу частотной характеристики разомкнутой системы, т.е. по ее АФЧХ . Изменение аргумента вектора F(jω) будет равно нулю, если АФЧХ разомкнутой системы не охватывает точку с координатами (– 1, j0). Если система содержит r интегрирующих звеньев, число r которых определяет степень астатизма системы, то начальное значение фазовой частотной характеристики равно –r , а амплитудной частотной – бесконечности.

1. Если разомкнутая система устойчива, то для устойчивости замкнутой системы необходимо и достаточно, чтобы АФЧХ разомкнутой системы не охватывала точку с координатами (–1, j0).

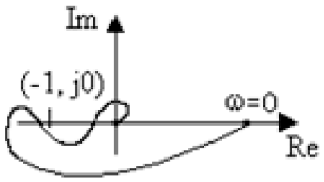
2. Если разомкнутая система неустойчива, то для устойчивости замкнутой системы необходимо и достаточно, чтобы АФЧХ разомкнутой системы охватывала точку с координатами (–1, j0) и при изменении частоты от 0 до ∞ оборачивалась вокруг нее против часовой стрелки m раз, где m – число полюсов разомкнутой системы с положительной вещественной частью.{число положительных (сверху вниз) переходов характеристики через ось абсцисс левее точки (–1) равняется числу отрицательных переходов (снизу вверх)

Абсолютно устойчивая (неустойчивость может наступить только при увеличении общего коэффициента передачи) Условно устойчивая (Неустойчивой такая система может быть как при увеличении, так и при уменьшении общего коэффициента передачи).На границе устойчивости/нейтральная. Не устойчивая.

Условно устойчивая



Графики соответствуют абсолютно устойчивой, нейтральной и неустойчивой системам.



АФЧХ статических разомкнутых систем

Вопрос 2. Сети встречного распространения. Слой Кохоненна, слой Гроссберга

Сети встречного распространения (Counterpropagation Networks) — это архитектура искусственных нейронных сетей, сочетающая принципы обучения с учителем и без учителя. Они состоят из двух основных слоев: слоя Кохоненна и слоя Гроссберга.

Слой Кохоненна

Слой Кохоненна выполняет функцию кластеризации данных без учителя. Он формирует карту признаков, группируя входные данные по схожести. Алгоритм работы слоя основан на поиске нейрона-победителя, веса которого ближе всего к входному вектору. Весовые коэффициенты этого нейрона и соседних нейронов корректируются в направлении входного вектора.

Слой Гроссберга

Слой Гроссберга используется для обучения с учителем. Он связывает результаты кластеризации (индекс нейрона-победителя в слое Кохоненна) с целевыми выходами. Этот слой корректирует веса на основе ошибки между целевым выходом и фактическим значением, что позволяет сети выполнять задачи регрессии или классификации.

Итог

Сети встречного распространения эффективно решают задачи, где необходимо предварительное структурирование данных (кластеризация), а затем — настройка на конкретные целевые данные. Такая архитектура обеспечивает адаптивность и высокую скорость обучения.

Вопрос 3. Процессорные ядра фирмы Xilinx. Особенности архитектуры процессорных ядер Picoblaze и Microblaze

Компания Xilinx разрабатывает процессорные ядра для встраиваемых систем, интегрируемые в ПЛИС. Два наиболее известных ядра — Picoblaze и Microblaze.

Picoblaze

Picoblaze — это простое 8-битное процессорное ядро с фиксированной архитектурой. Его особенности:

- Минимальный набор инструкций (около 40 команд).
- Жёстко фиксированная архитектура: 8-битные регистры, 8-битная шина данных.
- Отсутствие стека и поддержки прерываний.
- Используется для простых задач: управление портами ввода-вывода, обработка сигналов, выполнение простых алгоритмов.
- Компактный размер позволяет использовать Picoblaze в проектах с ограниченными ресурсами ПЛИС.

Microblaze

Microblaze — более сложное и универсальное 32-битное процессорное ядро с архитектурой Harvard.

- Конфигурируемая архитектура: пользователи могут выбирать набор команд, поддержку кэша, периферийных модулей.
- Поддержка прерываний, исключений, системного таймера.
- Используется для сложных приложений: цифровая обработка сигналов, сетевые протоколы, управление сложными системами.
- Поддерживает интеграцию с внешними интерфейсами через AXI-шину.

Сравнение

Picoblaze — простое ядро для базовых задач с минимальными требованиями к ресурсам, Microblaze — гибкий процессор для более сложных приложений, где требуется расширенный функционал.

Архитектура процессорных ядер Xilinx позволяет проектировать системы на кристалле (SoC), обеспечивая высокую степень интеграции и производительность.

Экзаменационный билет №9

Вопрос 1. Математический аппарат описания решетчатых функций. Теоремы и свойства Z-преобразования

Решетчатые функции — это функции, определенные на дискретных множественных структурах, например, на множествах, частично упорядоченных по определенному закону. В системах автоматического управления часто используются дискретные сигналы, которые удобно описывать с помощью таких функций, а их анализ и синтез выполняется с использованием Z-преобразования.

Теоремы и свойства Z-преобразования

Z-преобразование — основной инструмент анализа линейных дискретных систем. Основные свойства Z-преобразования:

1. **Линейность**: $Z\{ax_1[k] + bx_2[k]\} = aX_1(z) + bX_2(z)$
2. **Сдвиг по времени** (задержка на n шагов): $Z\{x[k-n]\} = z^{-n}X(z)$
3. **Сдвиг по времени влево** (ускорение): $Z\{x[k+n]\} = z^n(X(z) - x[0] - x[1]z^{-1} - \ldots - x[n-1]z^{-(n-1)})$
4. **Скалярное умножение по времени**: $Z\{a^k x[k]\} = X(z/a)$
5. **Теорема свертки**: $Z\{x[k] * y[k]\} = X(z)Y(z)$
6. **Теорема конечных значений**: $\lim_{k\to\infty} x[k] = \lim_{z\to 1}(1 - z^{-1})X(z)$
7. **Теорема начальных значений**: $x[0] = \lim_{z\to\infty} X(z)$

Z-преобразование используется для перехода из временной области в частотную и обратно, что позволяет анализировать устойчивость и поведение систем.

Вопрос 2. Алгоритм обучения по Хэббу

Алгоритм Хэбба — один из фундаментальных методов обучения в нейронных сетях, основанный на принципе: “Нейроны, которые возбуждаются одновременно, усиливают связь друг с другом”.

Формально правило Хэбба для обновления весов записывается так: $w_{ij}^{(new)} = w_{ij}^{(old)} + \eta \cdot x_i \cdot y_j$ где:

- w_{ij} — вес связи между нейроном i и нейроном j,
- η — коэффициент обучения (малое положительное число),
- x_i — значение входного сигнала нейрона i,
- y_j — значение выходного сигнала нейрона j.

Иными словами, веса усиливаются, если и входной, и выходной сигналы активны одновременно. Это правило положило начало развитию теорий обучения без учителя в нейронных сетях.

Характерные особенности обучения по Хэббу:

- Локальный характер обновления весов.
- Прямое отражение взаимной корреляции сигналов.
- Применимо для формирования карты признаков и обучения систем распознавания образов.

Метод Хэбба часто применяется в простых архитектурах и является основой для построения более сложных моделей обучения, таких как когнитроны и нейросетевые автоассоциаторы.

Вопрос 3. Организация памяти в системах с процессорными ядрами Nios II

Организация памяти в системах Nios II отличается гибкостью, так как память конфигурируется в зависимости от конкретного проекта.

Пространство адресов:

1. **Единое адресное пространство**:
    - 32-битное адресное пространство, позволяющее адресовать до 4 ГБ памяти.
    - Используется для программного кода, данных и периферийных устройств.
2. **Типы памяти**:
    - **Оперативная память (RAM)** — для хранения данных и выполнения программ.
    - **Постоянная память (ROM/Flash)** — для хранения программного кода и констант.
    - **Кэш-память** (в конфигурациях Nios II/f):
        - Инструкционный кэш (для ускорения загрузки команд).
        - Кэш данных (для ускорения операций с памятью).

Иерархия памяти:

1. **Локальная память (On-Chip Memory)**:
    - Встроенная память ПЛИС, доступная для быстрого доступа.
2. **Внешняя память (Off-Chip Memory)**:
    - Подключается через контроллер памяти.

Организация доступа к памяти:

1. **Прямой доступ (Direct Memory Access, DMA)**:
    - Используется для передачи данных между периферией и памятью без загрузки процессора.
2. **Блочный доступ**:
    - Поддержка объединения операций чтения/записи для повышения производительности.

Экзаменационный билет №10

Вопрос 1. Построение переходных процессов. Разложение в ряд Лорана

Построение переходных процессов

Переходный процесс в системе — это изменение состояния системы во времени после воздействия на неё внешнего возмущения или начальных условий. В теории автоматического управления и радиотехнике построение переходных процессов тесно связано с методами анализа линейных систем, такими как метод обратного преобразования Лапласа.

Для построения переходного процесса $y(t)$ системы, заданной передаточной функцией $W(s)$, используют следующие шаги: 1. Находят изображение отклика в области Лапласа:

$$Y(s) = W(s) \cdot X(s)$$

где $X(s)$ — изображение входного сигнала. 2. Выполняют разложение $Y(s)$ на простые дроби, чтобы затем воспользоваться таблицами преобразования Лапласа. 3. На основе разложения находят выражение $y(t)$ через обратное преобразование Лапласа.

Особенно важен учёт полюсов передаточной функции: каждый полюс порождает слагаемое в виде экспоненты или затухающей синусоиды, что и формирует форму переходного процесса.

Разложение в ряд Лорана

Разложение в ряд Лорана применяется в комплексном анализе для функций, имеющих особенности (особые точки). Оно позволяет выразить функцию $f(s)$ в виде:

$$f(s) = \sum_{n=-\infty}^{\infty} a_n (s - s_0)^n$$

где s_0 — точка, вокруг которой строится разложение.

Для анализа переходных процессов разложение в ряд Лорана используется при нахождении обратного преобразования Лапласа через метод вычетов: - В случае простых полюсов на комплексной плоскости s разложение функции по ряду Лорана позволяет вычислить вычет, а значит, и находить обратное преобразование. - Каждое слагаемое ряда Лорана в области s соответствует определённому элементу во временной области t после преобразования Лапласа.

Таким образом, разложение в ряд Лорана позволяет связать поведение функции вблизи особых точек с переходным процессом во временной области.

Вопрос 2. Неокогнитрон: сходство с когнитроном и отличие от него, структура сети

Сходство с когнитроном

Неокогнитрон — это многослойная иерархическая нейронная сеть, разработанная Кунихико Фукусимой в 1980 году для распознавания образов. Она является развитием идеи когнитрона, предложенного Фукусимой ранее. Оба типа сетей: - Моделируют обработку информации, подобную обработке в зрительной коре мозга. - Имеют слоистую архитектуру: чередование слоёв объединения (S-слоёв) и слоёв конкуренции/выборки (C-слоёв). - Используют локальные рецептивные поля: каждый нейрон обрабатывает ограниченную часть входного изображения.

Отличия от когнитрона

Основные отличия неокогнитрона от когнитрона: - В неокогнитроне улучшены механизмы обучения: вместо простого обучения по типу Хэбба применяется механизм с локальными правилами обучения и самонастройкой. - Вводится более сложная структура слоёв: - S-слои выполняют операцию свёртки с обучаемыми ядрами (аналог фильтров), усиливая локальные признаки. - C-слои выполняют операцию подвыборки (пулинга), повышая инвариантность к сдвигам. - Неокогнитрон более устойчив к сдвигам, искажениям и частичной потере информации по сравнению с когнитроном.

Архитектура неокогнитрона

Архитектура неокогнитрона напоминает современные сверточные нейронные сети и состоит из повторяющихся блоков: 1. **S-слои (Simple cells)** — извлечение локальных признаков из входных данных через свёртку. 2. **C-слои (Complex cells)** — агрегация выходов S-слоёв, подавление слабых реакций, инвариантность к позициям. 3. Многократное чередование S- и C-слоёв позволяет формировать сложные признаки из простых, например, линии, углы, фигуры. 4. Последний слой — классификационный, определяющий принадлежность входного изображения к одному из классов.

Таким образом, неокогнитрон является прототипом современных сверточных сетей и показывает, как можно построить систему распознавания образов, устойчивую к шумам и искажениям.

Вопрос 3. Особенности системы команд процессорного ядра Nios II фирмы Altera

Система команд Nios II отличается простотой и эффективностью, что соответствует принципам RISC-архитектуры.

Основные особенности:

1. **Формат инструкций**:
    - Все инструкции фиксированной длины (32 бита).
    - Поддерживаются три формата инструкций: R-тип (регистр-регистр), I-тип (регистр-немедленное значение) и J-тип (прыжки).

2. **Набор команд**:
    - **Арифметические операции**: сложение, вычитание, умножение, деление.
    - **Логические операции**: AND, OR, XOR, NOT.
    - **Сдвиги**: логические и арифметические влево и вправо.
    - **Переходы и ветвления**: условные (BEQ, BNE) и безусловные (JMP).
    - **Операции с памятью**: загрузка (LOAD) и сохранение (STORE).
3. **Регистры**:
    - Всего 32 регистра общего назначения (R0–R31), где R0 всегда равен 0.
    - Специальные регистры: регистр состояния (status), регистр возврата (ra).
4. **Особенности реализации**:
    - В некоторых конфигурациях доступны аппаратные ускорители для работы с плавающей запятой.
    - Поддержка пользовательских инструкций (custom instructions), позволяющая расширять функциональность.

Пример кода на языке ассемблера Nios II:

```
addi r8, r0, 10    // Загрузить значение 10 в регистр r8
addi r9, r0, 20    // Загрузить значение 20 в регистр r9
add r10, r8, r9    // Сложить r8 и r9, результат в r10
beq r10, r0, end   // Если r10 == 0, перейти к метке end
end:
```

Экзаменационный билет №11

Вопрос 1. Синтез корректирующих устройств ЦАС методом расчета по аналоговому прототипу

Принцип метода аналогового прототипа

Синтез корректирующих устройств цифровых автоматических систем (ЦАС) методом аналогового прототипа основан на использовании известных аналоговых фильтров и корректирующих устройств в качестве исходной модели. Основная идея заключается в том, чтобы: 1. Спроектировать корректирующее устройство в аналоговой форме, используя классические методы (например, частотный метод, корневой метод). 2. Затем преобразовать аналоговую систему в цифровую, используя подходящее преобразование (обычно билинейное или импульсно-инвариантное).

Этапы синтеза

1. **Проектирование аналогового прототипа**:
   - Разрабатывается передаточная функция аналогового корректирующего устройства $G_a(s)$ на основе требований к системе (устойчивость, точность, быстродействие).
   - Например, могут использоваться фильтры Баттерворта, Чебышева, корректирующие устройства типа ПИД-регуляторов.
2. **Выбор метода дискретизации**:
   - Обычно используется билинейное w-преобразование:
     $$s = \frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}}$$
     где T — шаг дискретизации.
   - При необходимости может быть применено импульсно-инвариантное преобразование.
3. **Преобразование передаточной функции**:
   - Аналоговый прототип $G_a(s)$ преобразуется в цифровую передаточную функцию $G_d(z)$ по выбранной формуле.
   - При этом важно учесть возможное искажение частотного диапазона (используется предискажение частоты).
4. **Получение разностного уравнения**:
   - На основе полученной $G_d(z)$ формируется разностное уравнение, реализующее работу корректирующего устройства в ЦАС.

Преимущества метода

- Возможность использования богатого опыта аналогового проектирования.
- Сохранение характеристик аналоговой системы (например, устойчивости, формы переходного процесса) в цифровом варианте.

Вопрос 2. Нейронная сеть Хэмминга

Основная идея сети Хэмминга

Нейронная сеть Хэмминга — это разновидность сетей сопоставления образов (ассоциативной памяти), предназначенная для классификации и распознавания входных шаблонов. Сеть названа в честь американского математика Ричарда Хэмминга и основана на использовании метрики Хэмминга для оценки близости между векторами.

Архитектура сети

Сеть Хэмминга обычно имеет **двухслойную архитектуру**: 1. **Первый слой (линейный слой)**: - Выполняет вычисление сходства между входным вектором X и эталонными векторами W_i, используя скалярное произведение или косинусное сходство. - Весовые коэффициенты первого слоя задаются заранее и равны обучающим образцам. - Результаты вычислений первого слоя подаются на второй слой.

2. **Второй слой (нелинейный слой)**:
    - Реализует механизм подавления побочных реакций (механизм подавления конкурентов, lateral inhibition).
    - На выходе остаётся активированным только один нейрон, соответствующий наиболее близкому образцу.
    - Обычно используется правило максимального отклика.

Принцип работы

- На вход сети подаётся вектор X.
- Первый слой вычисляет “сходство” S_i между входным вектором и каждым эталоном:

$$S_i = \sum_j W_{ij} X_j$$

- Второй слой подавляет все отклики, кроме максимального.
- Победивший нейрон указывает на класс, к которому относится входной вектор.

Особенности

- Сеть Хэмминга обучается **однократно**, просто запоминая эталоны.
- Быстрая классификация: процесс распознавания требует одного прямого прохода через сеть.
- Основная область применения — задачи классификации и распознавания образов (например, текстовые шаблоны, сигналы, образы).

Отличие от других сетей

В отличие от многослойных персептронов, сеть Хэмминга не требует многократного обучения через обратное распространение ошибки. Её задача — находить ближайший эталон в памяти на основе заданной метрики (метрики Хэмминга или косинусной меры).

Вопрос 3. Проектирование на основе языков описания аппаратных средств

Языки описания аппаратных средств (HDL)

Языки HDL (Hardware Description Languages) предназначены для описания структуры и поведения цифровых устройств. Основными языками являются Verilog и VHDL. Они позволяют описывать как комбинационные, так и последовательностные схемы на уровне абстракции, близком к архитектуре устройства.

Основные этапы проектирования

1. **Описание схемы**:
    - Поведенческое описание (behavioral): описание функциональности без детализации архитектуры.
    - Структурное описание (structural): описание взаимосвязей между модулями.
2. **Синтез схемы**:
    - Преобразование HDL-кода в сетевой граф (netlist), который отражает использование аппаратных ресурсов FPGA.
3. **Функциональная симуляция**:
    - Проверка правильности работы схемы до этапа синтеза.
    - Используются инструменты ModelSim, QuestaSim и аналогичные.
4. **Размещение и трассировка**:
    - Размещение логических блоков и маршрутизация соединений на кристалле FPGA.
5. **Анализ временных характеристик**:
    - Проверка выполнения временных ограничений для корректной работы устройства.
6. **Программирование устройства**:
    - Генерация конфигурационного файла и загрузка его в FPGA.

Преимущества проектирования на основе HDL

- **Ускорение разработки**: позволяет автоматизировать многие процессы проектирования.
- **Портативность**: HDL-код может быть использован для различных FPGA.
- **Модульность**: поддержка иерархии позволяет разрабатывать сложные проекты, используя простые модули.

```
module and_gate (
    input wire a,
    input wire b,
    output wire y
);
    assign y = a & b;
endmodule
```

Проектирование на основе HDL позволяет разработчикам создавать сложные цифровые устройства, используя гибкие инструменты для симуляции, синтеза и оптимизации логики.

Экзаменационный билет №12

Вопрос 1. Использование билинейного w-преобразования. Использование псевдочастоты

Билинейное w-преобразование

Билинейное преобразование — это метод перехода от непрерывной к дискретной передаточной функции в цифровой обработке сигналов и системах управления. Оно позволяет преобразовать аналоговую передаточную функцию $H(s)$ в цифровую $H(z)$, сохраняя основные характеристики системы (например, устойчивость и характер переходных процессов).

Формула билинейного преобразования:

$$s = \frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}}$$

где: - s — комплексная переменная в области Лапласа, - z — комплексная переменная Z-области, - T — шаг дискретизации.

Это преобразование нелинейно и искажает частотную ось: линейная шкала в аналоговой системе сжимается к низким частотам в цифровой системе. Для компенсации этого эффекта используется техника **предварительной подстановки частоты** (frequency pre-warping).

Псевдочастота

При билинейном преобразовании возникает так называемая **псевдочастота** (Ω), которая отражает несоответствие между аналоговой частотой ω_a и цифровой ω_d. Связь между ними:

$$\Omega = 2 \cdot \tan\left(\frac{\omega_d T}{2}\right)$$

где: - Ω — аналоговая частота в области Лапласа $s = j\Omega$, - ω_d — частота в цифровой области (в радианах на отсчет), - T — шаг дискретизации.

Использование псевдочастоты позволяет корректно сопоставлять характеристики аналоговой и цифровой систем, а также более точно рассчитывать коэффициенты фильтров и корректирующих устройств.

Вопрос 2. Проблема переобучения алгоритма обратного распространения ошибки

Обучение нейросети методом обратного распространения ошибки

Алгоритм обратного распространения ошибки (backpropagation) — это метод оптимизации весов в многослойных нейросетях. Сеть обучается на примерах: входные

данные подаются на вход сети, вычисляется ошибка между выходом сети и целевым значением, затем веса корректируются с помощью градиентного спуска.

Проблема переобучения

Переобучение (overfitting) возникает, когда нейросеть слишком хорошо подгоняет модель под обучающие данные, включая шум и случайные отклонения, но теряет способность обобщать информацию для новых данных. Это приводит к ухудшению качества работы сети на тестовых данных, которые она раньше не видела.

Основные причины переобучения: - Слишком сложная модель (много нейронов и слоёв) по сравнению с количеством обучающих данных. - Отсутствие регуляризации. - Длительное обучение без контроля за ошибкой на валидационной выборке. - Наличие шума и ошибок в данных.

Методы борьбы с переобучением

- **Регуляризация**: добавление штрафов за слишком большие веса (например, L_2 или L_1 регуляризация).
- **Раннее остановка (early stopping)**: прекращение обучения, когда ошибка на валидационной выборке перестаёт уменьшаться.
- **Dropout**: случайное отключение нейронов на тренировке для предотвращения избыточного обучения связей.
- **Увеличение объёма обучающей выборки**: помогает сети учиться на большем количестве примеров и улучшать обобщающие способности.
- **Кросс-валидация**: регулярная проверка качества на невидимых данных.

Заключение

Проблема переобучения — ключевая трудность в обучении нейросетей. Для создания надёжных моделей важно контролировать процесс обучения, использовать регуляризацию и проверять обобщающую способность сети на тестовых данных.

Вопрос 3. Маршрут проектирования SOPC и возможности САПР Quartus

Маршрут проектирования SOPC (System on a Programmable Chip):

Проектирование систем с процессорами Nios II осуществляется через специализированный инструмент Platform Designer (ранее Qsys), входящий в состав среды Quartus.

1. **Этапы проектирования**:
    - **Создание системы**:
        - Определение процессора Nios II, добавление периферийных устройств.
        - Настройка соединений между компонентами через шины Avalon.
    - **Настройка параметров**:

        - Установка тактовой частоты, ширины шин и размеров памяти.
    - **Генерация HDL-кода**:
        - Автоматическая генерация кода на Verilog/VHDL для всей системы.
    - **Синтез и компиляция**:
        - Оптимизация и загрузка проекта в ПЛИС.
2. **Создание программы**:
    - **Компиляция кода**:
        - Написание кода на языке C и его компиляция
    - **Загрузка и отладка программы**:
        - Загрузка скомпилированный бинарный файл (.elf) в память процессора Nios II через JTAG и отладка с помощью Eclipse IDE
3. **Поддержка тестирования**:
    - Создание тестовых прошивок для отладки с использованием встроенных средств.
    - Интеграция с инструментами анализа производительности (SignalTap II).

Возможности Quartus для разработки с Nios II:

1. **Автоматическая генерация системы**:
    - Упрощённый процесс проектирования сложных систем благодаря удобному интерфейсу Platform Designer.
2. **Поддержка пользовательских модулей**:
    - Интеграция собственных аппаратных блоков через шины Avalon.
3. **Среда для разработки программы**:
    - Наличие интегрированной среды разработки Eclipse IDE для генерации проекта и создания программы.
4. **Инструменты отладки**:
    - Использование встроенного отладчика для пошагового выполнения программ Nios II.
5. **Оптимизация производительности**:
    - Анализ использования ресурсов и автоматическая настройка параметров синтеза.

Таким образом, Quartus и SOPC Designer предоставляют мощный инструмент для разработки настраиваемых систем с процессорами Nios II, что делает их незаменимыми для задач создания встраиваемых решений.

Экзаменационный билет №13

Вопрос 1. Рекурсивные и нерекурсивные корректирующие фильтры ЦАС

Корректирующие фильтры в системах автоматического управления (САУ) используются для изменения динамических свойств системы, улучшения устойчивости и точности регулирования. В зависимости от структуры различают **рекурсивные** (с обратной связью) и **нерекурсивные** (без обратной связи) фильтры.

Нерекурсивные (конечные импульсные характеристики, FIR)

Нерекурсивные фильтры описываются разностным уравнением, в котором выход в текущий момент времени зависит только от текущего и предыдущих значений входа:

$$y(k) = \sum_{i=0}^{N} b_i \, x(k-i)$$

где $x(k)$ — входной сигнал, $y(k)$ — выходной сигнал, b_i — коэффициенты фильтра.

Особенности: - Простота реализации. - Гарантированная устойчивость, так как нет обратной связи. - Ограниченная возможность моделирования сложных динамических свойств. - Применяются для задач, где важна линейная фазовая характеристика (например, в цифровой обработке сигналов).

Рекурсивные (бесконечные импульсные характеристики, IIR)

Рекурсивные фильтры используют обратную связь, и выходной сигнал зависит как от входных данных, так и от предыдущих выходов:

$$y(k) = \sum_{i=0}^{M} b_i \, x(k-i) - \sum_{j=1}^{N} a_j \, y(k-j)$$

Особенности: - Более компактное представление сложных фильтров (по сравнению с нерекурсивными). - Возможность реализации фильтров с резонансными свойствами и острыми пиками на АЧХ. - Возможность возникновения устойчивости или неустойчивости в зависимости от выбора коэффициентов a_j. - Широко используются для реализации корректирующих звеньев в ЦАС, например, ПИД-регуляторов.

Заключение

- **Нерекурсивные фильтры** — проще, всегда устойчивы, но менее гибкие.
- **Рекурсивные фильтры** — сложнее, требуют контроля устойчивости, но позволяют реализовать более широкий класс фильтров.

Вопрос 2. Двухслойность персептрона

Персептрон — это классическая модель искусственного нейрона, предложенная Ф. Розенблаттом в 1958 году для распознавания образов и решения задач классификации. Основной особенностью персептрона является его **двухслойная архитектура**.

Архитектура двухслойного персептрона

Персептрон состоит из двух слоев: 1. **Входной слой**: - Каждый нейрон входного слоя просто передаёт значения входных признаков (векторов данных) в следующий слой. - Входы могут быть нормализованными данными, бинарными или действительными значениями.

2. **Слой вычислительных нейронов (выходной слой)**:
    - Каждый нейрон этого слоя вычисляет взвешенную сумму входов:
    $$y = \varphi\left(\sum_{i=1}^{n} w_i\, x_i + b\right)$$
    где x_i — входные данные, w_i — веса, b — смещение, φ — функция активации (обычно ступенчатая или сигмоидальная).
    - Выходы нейронов определяют класс входного вектора.

Ограничения и возможности

- **Двухслойный персептрон** (один слой весовых связей) способен решать только линейно разделимые задачи. Например, он может классифицировать точки, разделённые прямой, но не способен решить задачу XOR.
- Для решения нелинейно разделимых задач требуется добавление скрытых слоёв, что приводит к многоуровневым сетям.

Итог

- Двухслойный персептрон — базовый элемент нейронных сетей, важный для понимания принципов обучения и классификации.
- Его возможности ограничены линейными задачами, поэтому дальнейшее развитие нейросетей привело к созданию многослойных персептронов и более сложных архитектур.

Вопрос 3. Архитектура процессорных ядер Nios II, основные конфигурации систем

Процессорное ядро Nios II, разработанное фирмой Altera (ныне Intel), — это универсальное 32-битное RISC-ядро, которое может быть интегрировано в ПЛИС для создания настраиваемых систем-на-кристалле (SoC).

Основные характеристики архитектуры:

1. **RISC-архитектура**:
    - Nios II использует простую и эффективную архитектуру с фиксированной длиной инструкции (32 бита).

2. **Модульность и настраиваемость**:
    - Ядро поддерживает множество настроек, таких как число регистров, аппаратное умножение и деление, уровень производительности и объём используемой памяти.
3. **Конвейерная обработка**:
    - Реализован трехступенчатый конвейер (fetch, decode, execute) для повышения производительности.

Основные конфигурации Nios II:

1. **Nios II/e (Economy)**:
    - Минимальный размер ядра с минимальными аппаратными ресурсами.
    - Подходит для простых приложений, где критична экономия места.
2. **Nios II/s (Standard)**:
    - Сбалансированная конфигурация, обеспечивающая хорошую производительность при умеренном использовании ресурсов.
3. **Nios II/f (Fast)**:
    - Максимальная производительность за счёт включения таких функций, как аппаратное умножение, кэш инструкций и данных.

Встроенные периферийные устройства:

Nios II легко интегрируется с различными периферийными компонентами, такими как UART, SPI, таймеры, а также с пользовательскими логическими блоками, разработанными в HDL.

Экзаменационный билет №14

Вопрос 1. Определение устойчивости ЦАС по логарифмическим частотным характеристикам

Устойчивость систем автоматического управления (САУ) может быть определена с помощью анализа логарифмических частотных характеристик (ЛАЧХ), которые включают амплитудно-частотную характеристику (АЧХ) и фазо-частотную характеристику (ФЧХ). Этот метод основан на частотном критерии Найквиста и широко применяется при проектировании и анализе систем управления, особенно когда требуется визуальная оценка запаса устойчивости системы.

Основные принципы

Для определения устойчивости ЦАС по ЛАЧХ используется понятие **запаса устойчивости**: - **Запас по амплитуде (Азап)** — отношение фактического усиления системы к такому усилению, при котором система становится на грани устойчивости. В графическом виде это вертикальное расстояние (в дБ) между АЧХ и линией 0 дБ в точке, где фазовый сдвиг достигает –180°. - **Запас по фазе (фзап)** — угол между фазовой характеристикой и линией –180° в точке пересечения АЧХ с линией 0 дБ. Это горизонтальное расстояние на ФЧХ.

Критерий устойчивости по ЛАЧХ

Система считается устойчивой, если: 1. АЧХ системы пересекает уровень 0 дБ на такой частоте, где фазовый сдвиг меньше –180° (т.е. фзап > 0). 2. На частоте, где фазовый сдвиг достигает –180°, АЧХ должна находиться ниже 0 дБ (Азап > 0).

Эти условия позволяют оценить, насколько устойчива система к изменениям параметров, и избежать колебательного или неустойчивого поведения.

Преимущества метода

- Позволяет проектировать системы с заданными запасами устойчивости.
- Удобен для работы с системами, содержащими неизменяемую часть (например, технологический процесс).
- Визуализация частотных свойств помогает понять поведение системы на разных частотах.

Вопрос 2. Радиальная нейронная сеть и ее архитектура

Радиальная нейронная сеть (RBF-сеть) — это разновидность искусственных нейронных сетей, которая широко используется для решения задач классификации, регрессии, аппроксимации и интерполяции данных. Она названа так, потому что в качестве функции активации используется радиальная базисная функция, обычно зависящая от расстояния между входным вектором и центром нейрона.

Архитектура RBF-сети

Структура RBF-сети обычно включает три слоя: 1. **Входной слой** — принимает входные данные, каждый нейрон связан с одним компонентом входного вектора. 2. **Скрытый слой** — содержит нейроны с радиальными базисными функциями, например, гауссовыми. Каждый нейрон скрытого слоя вычисляет значение функции активации:

$$\phi(x) = \exp\left(-\frac{\| x - c_i \|^2}{2\sigma^2}\right)$$

где x — входной вектор, c_i — центр радиального нейрона, σ — параметр ширины функции. 3. **Выходной слой** — линейно комбинирует отклики скрытого слоя для формирования итогового ответа сети.

Особенности

- RBF-сеть обладает высокой скоростью обучения по сравнению с многослойным персептроном.
- Каждый скрытый нейрон локально реагирует на входные данные, что делает сеть устойчивой к шуму.
- Архитектура RBF хорошо подходит для аппроксимации сложных нелинейных зависимостей и задач классификации.

Вопрос 3. ПЛИС типа FPGA фирмы Xilinx семейств Artix-7, Kintex-7, Virtex-7, Zynq-7000

Программируемые логические интегральные схемы (ПЛИС, FPGA) фирмы Xilinx серии 7-го поколения представляют собой мощные платформы для создания цифровых устройств с гибкой архитектурой. Рассмотрим особенности каждого семейства:

Artix-7

Artix-7 — это семейство FPGA, ориентированное на приложения с низким энергопотреблением и стоимостью. Основные особенности: - Энергоэффективная архитектура на основе 28-нм техпроцесса. - Поддержка базовых функций цифровой логики, включая DSP-блоки, PLL и блоки памяти. - Часто используется в портативных устройствах, обработке сигналов, коммуникационном оборудовании.

Kintex-7

Kintex-7 — средний класс FPGA, сочетающий относительно низкое энергопотребление и высокую производительность: - Поддержка высокоскоростных интерфейсов, таких как PCIe и трансиверов до 12.5 Гбит/с. - Увеличенное количество логических элементов и ресурсов DSP по сравнению с Artix-7. - Используется в задачах обработки видео, связи и встроенных системах.

Virtex-7

Virtex-7 — высокопроизводительное семейство FPGA для задач, требующих максимальной вычислительной мощности: - Большое количество логических ячеек, DSP-блоков, поддержка многогигабитных интерфейсов (до 28 Гбит/с). - Применяется в телекоммуникационных системах, центрах обработки данных, сложных вычислительных системах.

Zynq-7000

Zynq-7000 — уникальная платформа, сочетающая в одном кристалле FPGA-логику и процессорное ядро ARM Cortex-A9: - Позволяет объединить гибкость программируемой логики и возможности процессора общего назначения. - Используется для создания систем на кристалле (SoC) для встраиваемых приложений, робототехники, обработки изображений, сетевых устройств.

Заключение

Выбор семейства зависит от требований задачи: - **Artix-7** — бюджетные решения с низким энергопотреблением. - **Kintex-7** — высокоскоростная обработка данных. - **Virtex-7** — задачи с экстремальной производительностью. - **Zynq-7000** — объединение процессорной и логической частей в едином чипе для гибких систем.