Таблица 4 – Сила осциллятора $Q_y(0,0)$ электронного перехода в зависимости от двугранных углов φ_5 и φ_{15} между плоскостями макроцикла и нитрогрупп

φ_5, °	φ_{15}, °				
	0	30	46	60	90
0	643.4	635.3	629.4	623.3	612.0
30	635.0	629.2	625.1	620.3	609.6
46	629.2	625.1	622.4	618.5	608.3
60	623.3	620.5	618.7	615.6	605.6
90	611.8	609.6	608.2	605.4	594.4
120	623.1	620.3	618.7	615.6	605.8
160	639.4	632.6	627.7	622.3	611.5
180	643.3	635.2	629.4	623.2	612.0

Таким образом, полученные результаты однозначно указывают на существенную зависимость формы спектров поглощения от двугранных углов между плоскостью макроцикла и плоскостями нитрогрупп.

ЛИТЕРАТУРА

1. Гладков Л.Л., Крук Н.Н. Спектральные проявления модуляции энергии электронных орбиталей макроцикла порфина при вращении NO_2-заместителя // Журн. прикл. спектр. – 2024. – Т. 91. – № 5. – С. 623–629.

УДК 004.4-048 (075.8)

Н.А. Жиляк, доц. (БГТУ, г. Минск)

ЯЗЫК JAVASCRIPT ДЛЯ СИНТЕЗА ВЫЧИСЛИТЕЛЬНЫХ СХЕМ НА ОСНОВЕ АЛГОРИТМОВ ОПТИМИЗАЦИИ

Век цифровых технологий требует от нас постоянного улучшения и оптимизации вычислительных процессов. Одним из ключевых аспектов в этой области является проектирование вычислительных схем, которые могут эффективно выполнять определенные задачи. В данной статье мы рассмотрим основные принципы проектирования вычислительных схем, методы оптимизации и их применение в различных областях [1, 2].

Представим некоторые этапы проектирования вычислительных схем:

1. Определение задачи. Проектирование вычислительной схемы начинается с четкого определения задачи. Это может быть как простая логическая функция, так и сложные алгоритмы обработки данных. На этом этапе важно учитывать требования к производительности, надежности и энергопотреблению.

2. Выбор архитектуры. Существует множество архитектур для

проектирования вычислительных схем, каждая из которых имеет свои преимущества и недостатки: комбинаторные схемы. Они принимают входные данные и выдают выходные данные без хранения состояния. Например, так работают мультиплексоры. Также схемы с памятью могут хранить состояние, что позволяет им выполнять более сложные операции. Примерами могут выступать триггеры и регистры. Параллельные схемы могут обрабатывать несколько задач одновременно, что значительно увеличивает производительность.

3. Моделирование схемы. На этом этапе проектировщик создает модель схемы с использованием языков описания аппаратуры (HDL), таких как VHDL или Verilog. Моделирование позволяет проверить функциональность схемы до ее физической реализации.

Оптимизация вычислительных схем может быть выполнена с помощью различных подходов и методов:

1. Логическая оптимизация. Этот подход включает упрощение логических выражений, что позволяет уменьшить количество используемых логических элементов. Методы которые используют применение карт Карно или алгоритм Куайна-МакКласки, широко используются для этой цели.

2. Оптимизация по времени. Оптимизация по времени направлена на уменьшение временных задержек в схеме. Это может быть достигнуто за счет изменения порядка выполнения операций или использования более быстрых логических элементов.

3. Оптимизация по мощности Энергопотребление становится критически важным, особенно в мобильных устройствах. Оптимизация по мощности включает в себя использование энергосберегающих компонентов и алгоритмов, которые минимизируют активное время работы схемы.

4. Оптимизация по площади. В некоторых случаях важно уменьшить физический размер схемы. Это может быть достигнуто путем использования более компактных логических элементов или путем интеграции нескольких функций в одном элементе.

Вычислительные схемы находят применение в различных областях, например, при разработке архитектуры компьютера. Оптимизированные схемы используются для создания процессоров, которые способны выполнять сложные вычисления быстрее и с меньшим потреблением энергии. При обработке сигналов в аудио- и видеотехнике оптимизированные схемы позволяют улучшить их качество. Если речь идет о разработке встраиваемых систем управления, например, микроконтроллеров, будут использоваться вычислительные схемы для управления устройствами и процессами.

Синтез вычислительных схем – это процесс проектирования и оптимизации схем, которые могут выполнять определенные вычисления. Использование JavaScript для этой задачи может быть интересным, особенно если вы хотите создать веб-приложение или инструмент для визуализации.

Вот пример подхода к синтезу вычислительных схем с использованием алгоритма оптимизации на JavaScript:

Шаг 1: Определение проблемы. Сначала нужно определить, какую задачу вы хотите решить. Например, это может быть минимизация логических функций или оптимизация маршрутов в графе.

Шаг 2: Алгоритм оптимизации. Вы можете использовать различные алгоритмы оптимизации, такие как генетические алгоритмы, алгоритмы муравьиной колонии или даже простые жадные алгоритмы. Для простоты мы рассмотрим жадный алгоритм.

Шаг 3: Реализация на JavaScript. Вот пример простого кода, который демонстрирует создание и оптимизацию вычислительной схемы:

JavaScript

```
class Node {
    constructor(value) {
        this.value = value;
        this.children = [];
    }
    addChild(child) {
        this.children.push(child);
    }
}
class Circuit {
    constructor() {
        this.nodes = [];
    }
    addNode(node) {
        this.nodes.push(node);
    }
    optimize() {
        // Пример жадной оптимизации: удаление неиспользуемых узлов
        this.nodes = this.nodes.filter(node => this.isUsed(node));
    }
    isUsed(node) {
        // Проверка, используется ли узел в других узлах
        return this.nodes.some(n => n.children.includes(node));
    }
    display() {
        // Отображение схемы (можно расширить для более сложного отображения)
        this.nodes.forEach(node => {
```

```
            console.log(Node:    ${node.value},    Children:
${node.children.map(child => child.value).join(', ')});
        });
    }
}
// Пример использования
const circuit = new Circuit();
const nodeA = new Node('A');
const nodeB = new Node('B');
const nodeC = new Node('C');
nodeA.addChild(nodeB);
nodeB.addChild(nodeC);
circuit.addNode(nodeA);
circuit.addNode(nodeB);
circuit.addNode(nodeC);
// Оптимизация схемы
circuit.optimize();
// Отображение результата
circuit.display();
```

Шаг 4: Визуализация. Для визуализации схемы можно использовать библиотеки, такие как D3.js, чтобы создать графическое представление ваших узлов и соединений.

Шаг 5: Расширение функциональности. В зависимости от ваших потребностей, вы можете добавлять больше функций, таких как: • Поддержка различных типов узлов (логические операции, арифметические операции и т.д.). Более сложные алгоритмы оптимизации. Интерфейс для ввода данных пользователем.

Синтез вычислительных схем с использованием JavaScript – это интересная задача, которая может быть реализована различными способами. Выбор алгоритма оптимизации и подхода к реализации будет зависеть от конкретных требований вашего проекта [1]. Проектирование вычислительных схем и их оптимизация – это сложный процесс, который требует глубоких знаний в области электроники и компьютерных наук. С учетом постоянно развивающихся технологий, важно оставаться в курсе новых методов и подходов к проектированию, чтобы создавать эффективные и надежные решения для современных задач.

ЛИТЕРАТУРА

1. Алгоритм синтеза схем многоканальных вторичных вычислений / Н.А. Жиляк, А.С. Кобайло // Труды БГТУ. №6(144), Физ.-мат. науки и информатика. – 2011. – С. 160–163.

2. Жиляк Н.А. / Методы синтеза вычислительных систем / Н.А. Жиляк // Автоматический контроль и автоматизация производственных процессов: материалы Междунар. науч.-техн. конф., Минск, 28–29 окт. 2009 г. – Минск: БГТУ, 2009. – С. 71–73.