СЕКЦИЯ 6. «ИНЖЕНЕРИЯ ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ»

УДК 004.414.2

Организация АЛУ с учетом архитектурных особенностей и оптимизаций

Д.В. Николаенко[*1], Я.И. Выростков[*2], Л.П. Володько[*3]

[*1] к.т.н, доцент, Донецкий национальный технический университет,
dv.nikolaenko@yandex.ru, SPIN-код: 9819-4763

[*2] магистрант, Донецкий национальный технический университет,
yarchikbroo129@gmail.com

[*3] к.т.н, доцент, Полесский государственный университет, г. Пинск, Республика Беларусь

Николаенко Д.В., Выростков Я.И., Володько Л.П. Организация АЛУ с учетом архитектурных особенностей и оптимизаций. *В статье рассмотрены подходы к организации АЛУ с учетом его функциональных и архитектурных особенностей. Описаны основные виды операций, выполняемых в АЛУ, включая операции над двумя операндами и операции сдвига. Особое внимание уделено оптимизации структуры АЛУ: сравниваются варианты использования отдельных схем для каждой операции, универсальной схемы и гибридного подхода, при котором операции разделяются на группы. Рассматривается роль сдвигов в алгоритмах умножения и деления, а также предлагается подход с выделением отдельного модуля для выполнения всех операций сдвига. Обсуждаются компромиссы между быстродействием, сложностью схемы и экономией аппаратных ресурсов, приводящие к выбору оптимальной архитектуры.*

Ключевые слова: *арифметико-логическое устройство, архитектура вычислительных устройств, оптимизация схемы, гибридный подход, быстродействие, упрощение архитектуры, аппаратные ресурсы.*

Введение

Для изучения принципов построения компьютерной архитектуры, особенностей функционирования отдельных элементов недостаточно продемонстрировать студентам детальные рисунки и даже подробное описание. Следует потратить время для непосредственной работы с вычислительной машиной, оснащенной пультом управления и различными индикаторами. Для данных нужд можно использовать либо эталонную машину [1], либо модель.

60-летний опыт кафедры компьютерной инженерии преподавания дисциплин, связанных с изучением проектирования аппаратных элементов вычислительных систем, показывает, что оптимальным способом организации процесса донесения и усвоения информации является применение специально разработанной виртуальной лаборатории [2-4]. Ее использование в учебном процессе удовлетворяет, с одной стороны, потребности преподавателей в средствах донесения информации и, с другой стороны, помогает студентам воспринимать новый материал, выполнять комплекс лабораторных и курсовых работ.

Арифметико-логическое устройство (АЛУ) является ключевым компонентом ядра вычислительных систем, обеспечивающим выполнение базовых арифметических и логических операций. От его архитектуры и организации зависят быстродействие, сложность реализации и эффективность работы всего устройства. В современной практике проектирования АЛУ часто требуется искать баланс между производительностью, экономией аппаратных ресурсов и универсальностью [5, 6].

Настоящая статья посвящена анализу различных подходов к организации АЛУ. Рассматриваются их преимущества и недостатки, а также приводятся рекомендации по выбору наиболее эффективной структуры для различных задач**.**

Основные операции в АЛУ

Арифметико-логическое устройство (АЛУ) служит для выполнения арифметических и логических преобразований над данными, или же операндами. Все выполняемые в арифметико-логическом устройстве операции являются логическими операциями (функциями), которые можно разделить на следующие группы [5]:

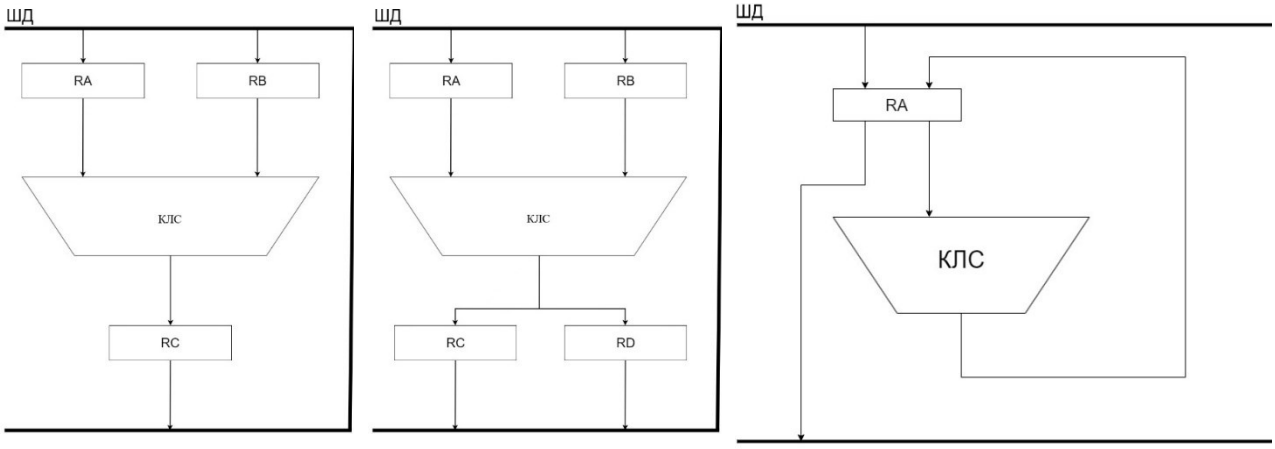
- операции двоичной арифметики для чисел с фиксированной точкой;
- операции двоичной арифметики для чисел с плавающей точкой;
- операции десятичной арифметики;
- операции специальной арифметики;
- логические операции;
- операции над алфавитно-цифровыми полями.

Современные компьютеры общего назначения обычно реализуют операции всех приведённых выше групп. К арифметическим операциям относятся сложение, вычитание, умножение и деление. Группу логических операций составляют операции дизъюнкция (логическое ИЛИ) и конъюнкция (логическое И) над операндами, сравнение кодов на равенство. АЛУ обладает набором операций, определяющим назначения устройства. Операции можно разделить на два вида:

- операции над двумя операндами – большинство арифметических и логических операций (сложение, вычитание, логические "И", "ИЛИ" и т.д.);

- операции над одним операндом – в основном сдвиги данных или инкремента/декремента.

Общая структура для операций над двумя операндами (рис. 1, а) содержит: регистры RA и RB для хранения исходных операндов; регистр результата RC и КЛС – комбинационно логическую схему, выполняющую определенную арифметическую или логическую операцию. Организация АЛУ для операции деления отличается наличием дополнительного регистра RD (рис. 1, б). Для операций с одним операндов достаточно использовать один регистр (рис. 1, в).



а) б) в)

Рисунок 1 – Структура АЛУ: а) для операций с двумя операндами; б) для операции деления; в) для операций с одним операндом

Подходы к реализации АЛУ

В зависимости от архитектурных предпочтений, мы можем организовать выполнение операций в АЛУ одним из трех способов.

Способ 1 - Отдельные схемы для каждой операции. В этом случае для каждой операции из набора выделяется отдельный функциональный блок, например, сумматор, делитель и т.д.

Преимущества:

- лаконичность архитектуры каждого отдельного блока;
- высокая скорость выполнения операций;

Недостатки:

- увеличение количества элементов схемы;
- повышенная стоимость реализации и увеличение занимаемого пространства на плате;

Способ 2 - Единая универсальная схема. В этом случае все операции выполняются с использованием одной универсальной схемы.

Преимущества:

- значительное сокращение количества элементов;
- компактность схемы;

Недостатки:

- снижение быстродействия;
- более сложная алгоритмическая структура;
- повышенная нагрузка на управляющий автомат.

Способ 3 - Гибридный вариант. При его использовании операции распределяются по нескольким группам, каждая из которых реализуется в отдельном модуле. Например, логические операции реализуются в одном модуле, арифметические – в другом, умножение и деление – в третьем.

Преимущества:

- частичное устранение недостатков предыдущих подходов;
- сохранение высокой производительности;
- оптимизация структуры алгоритмов;
- умеренное количество элементов на схеме;

Недостатки.

Гибридный подход сочетает сильные стороны специализированных и универсальных схем, но также неизбежно теряет их ярко выраженные преимущества. Он не обеспечивает ни максимального быстродействия, свойственного специализированным схемам, ни значительной компактности и экономии ресурсов, характерных для универсального решения. В результате архитектура получается компромиссной, без явных недостатков ни в одном аспекте [7, 8].

Разработка методов для реализации моделей АЛУ

Для реализации эмулятора АЛУ наиболее правильным кажется выбор объектно-ориентированного языка, т.к. все элементы можно описать как объекты, а затем настроить их соответствующее взаимодействие. Среди объектно-ориентированных языков довольно большой выбор – Java, C#, C++, Python и др. Каждый из них имеет ряд достоинств, а также устоявшиеся сферы применения. Так, C++ используют для высоконагруженных приложений и систем с высокими требованиями к производительности, C# применяют для разработки игр и приложений корпоративного сектора, Python обрел популярность в задачах машинного обучения и анализа данных. Одним из наиболее важных критериев является степень владения синтаксисом и стандартными библиотеками того или иного языка. По этому критерию в качестве языка программирования выбран C#. Среда программирования – традиционная Microsoft Visual Studio. C# предоставляет выбор платформы, на которой будет запускаться приложение. Среди других платформ выбрана .NET 8.0, в пользу которой говорит кроссплатформенность (персональный компьютер, веб-разработка, мобильные приложения) и долгосрочная поддержка платформы от производителя. Также в связке с C# базово поставляются несколько решений для проектирования пользовательского интерфейса – в частности, выбранное для текущего проекта – Windows Presentation Foundation. Выбор обусловлен актуальностью и гибким управлением размером элементов на форме.

В качестве примера операции над одним операндом приведена реализация метода Increment() (см. рис. 2).

```
Ссылок: 1
public void Increment()
{
    int TryOutput;
    if (!Int32.TryParse(ToStringBinFull(), System.Globalization.NumberStyles.BinaryNumber, null, out TryOutput))
        throw new Exception("ProgrammCounter: Недопустимые символы в строке счетчика команд!");
    TryOutput++;
    FromStringBinFull(Convert.ToString(TryOutput, 2).PadLeft(18, '0'));
}
```

Рисунок 2 – Реализация метода Increment()

В этом методе значение из памяти преобразуется в строковое, затем приводится к числу типа int, инкрементируется, обратно преобразуется в строковое, из которого перезаполняется память счетчика.

Схожая реализация выполнена и относительно метода Sum() класса EmulatorAccumulatorAdress. Оба слагаемых приводятся к целочисленному типу, выполняется сложение int, после чего результат помещается в память элемента (рис. 3).

```
Ссылок: 2
public void Sum(string Bin)
{
    if (Bin.Length > 32)
        throw new Exception("AdressAccumulator: Длина слагаемого превышает допустимую в аккумуляторе адреса!");
    if (Bin.Length < 32)
        Bin = Bin.PadLeft(32, '0');
    int iacum, iBin;
    if (!Int32.TryParse(ToStringBinFull(), System.Globalization.NumberStyles.BinaryNumber, null, out iacum))
        throw new Exception("AdressAccumulator: Недопустимые символы в строке аккумулятора!");
    if (!Int32.TryParse(Bin, System.Globalization.NumberStyles.BinaryNumber, null, out iBin))
        throw new Exception("AdressAccumulator: Недопустимые символы в строке регистра данных!");
    int rez = iacum + iBin;
    FromStringBin(Convert.ToString(rez, 2).PadLeft(32, '0'));
}
```

Рисунок 3 – Реализация метода Sum()

Разработка графического интерфейса эмулятора

Модель АЛУ является одной из составляющих эмулятора процессорного устройства. Графический интерфейс эмулятора реализован по аналогии с отладчиками ассемблерного кода. Такие программы обычно разбиты на несколько областей, каждая из которых отображает текущую информацию о регистрах, памяти и т.д.

Главное окно эмулятора имеет вид, показанный на рис. 4. Окно разбито на 5 областей: меню, левая панель (область команд), правая панель (область состояния регистров и флагов), нижняя панель (область просмотра памяти).

Элементы меню позволяет выполнять некоторые сервисные действия. Например, подменю Файл содержит кнопки, позволяющие сохранять текущее состояние эмулятора в файл, и также загружать его в эмулятор. Отдельно можно сохранить и загрузить значение памяти, т.е. в отрыве от эмулятора можно написать исполняемую программу с использованием форматов команд процессорного устройства.

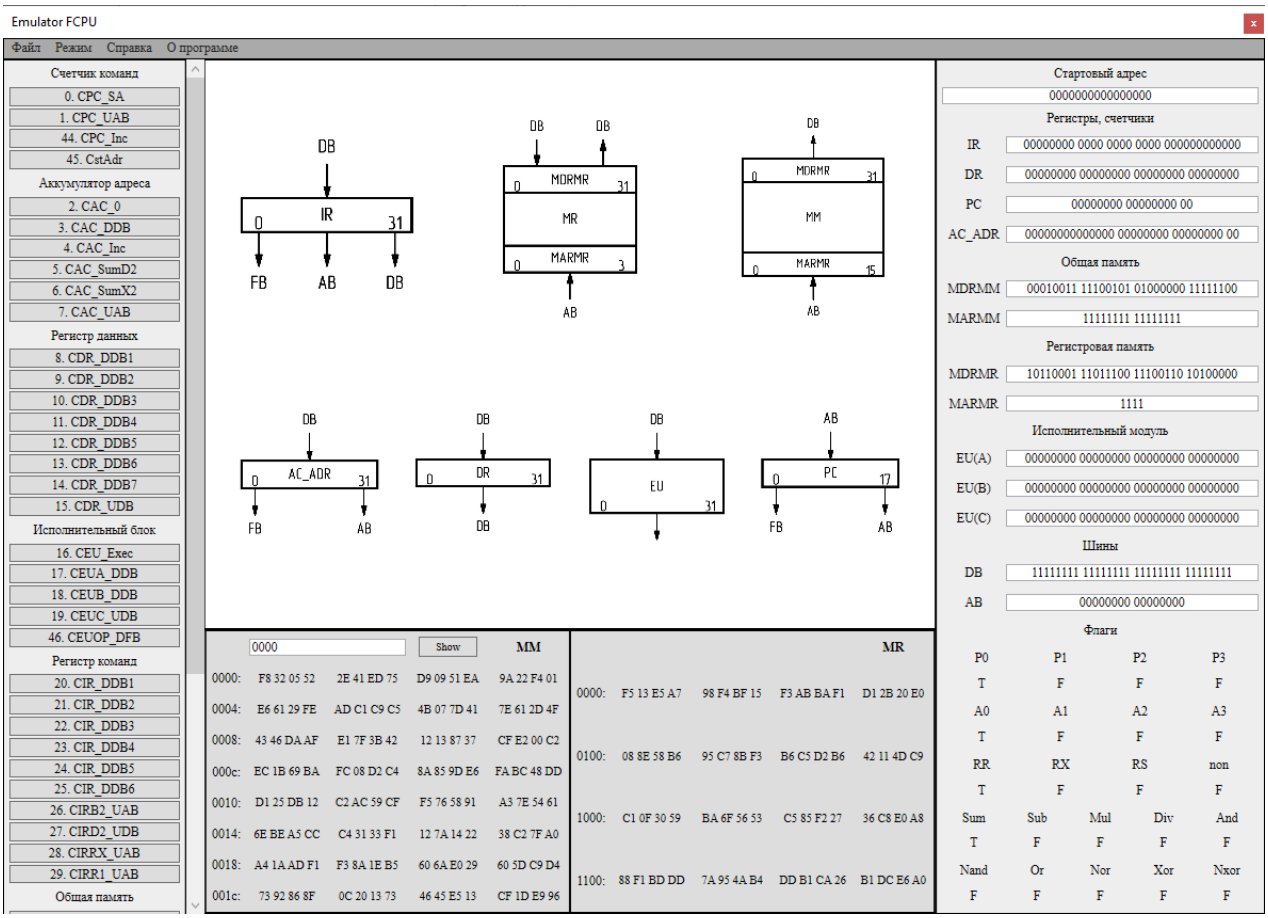


Рисунок 4 – Скриншот главного окна эмулятора

Подменю Режим позволяет выбрать ручной или автоматический режим. Автоматический режим запускает работу согласно граф-схеме алгоритма с некоторой задержкой, для отслеживания результатов. Его можно также назвать демонстрационным режимом. В этом режиме выполнение алгоритма можно поставить на паузу.

В ручном режиме предполагается самостоятельное исполнение микрокоманд путем нажатия на них в левой панели. На базе ручного режима можно построить выполнение лабораторной работы для отладки выведенного решения.

Подменю Справка позволяет дополнительно открыть окна со справочным материалом, которые содержат таблицу сигналов, флагов, полную функциональную схему и граф-схему алгоритма.

Пункт меню «О программе» выводит сообщение об авторе программы и ее назначении.

Левая панель – область команд, где можно применять определенную команду в ручном режиме.

Оптимизация работы сдвигов

Сдвиги играют важную роль в вычислительных алгоритмах, особенно в умножении и делении, где они могут быть частью последовательности микрокоманд. Чтобы сократить сложность модулей умножения и деления, можно вынести сдвиги в отдельный специализированный модуль. В этом случае данные передаются в модуль сдвига, там обрабатываются и возвращаются с изменением в требуемом направлении.

Такой подход упрощает конструкции других модулей, делая их компактнее и более универсальными. Однако он вводит дополнительную задержку из-за необходимости трех шагов: передачи данных в модуль, выполнения сдвига и возврата результата. Это может немного сказаться на быстродействии, но позволяет существенно упростить общую архитектуру вычислительного устройства.

Выводы

Проектирование АЛУ требует взвешенного подхода к выбору архитектуры, обеспечивающей баланс между производительностью, сложностью реализации и экономией аппаратных ресурсов.

Оптимизация работы АЛУ, включая вынесение операций сдвига в отдельный модуль, позволяет упростить архитектуру, однако сопровождается неизбежным снижением быстродействия. Итоговый выбор архитектуры должен основываться на конкретных требованиях системы, где приоритеты между производительностью, стоимостью и универсальностью будут определять оптимальное решение.

Литература

1. Кириллов, В. В. Архитектура базовой ЭВМ. – СПб: СПбГУ ИТМО, 2010. – 144 с.

2. Мостовая, В. А. Разработка виртуальной лаборатории для изучения архитектуры процессора [Текст] / В. А. Мостовая, О. А. Авксентьева, Р. В. Мальчева // Материалы VI Международной научно-технической конференции «Современные информационные технологии в образовании и научных исследованиях» (СИТОНИ-2019). – Донецк: ДонНТУ, 2019. - С.375-379.

3. Мальчева, Р. В. Разработка виртуальной лаборатории для изучения и моделирования архитектур процессорных элементов [Текст] / Р. В. Мальчева, О. А. Авксеньева // Программная инженерия: методы и технологии разработки информационно- вычислительных систем (ПИИВС-2016): сборник научных трудов I научно-практической конференции. 16-17 ноября 2016 г. – Донецк: ДонНТУ, 2016. - С. 102-108.

4. Мальчева, Р. В. Моделирование внутренних операций процессорных элементов / Р. В. Мальчева, Т. В. Завадская // Информатика и кибернетика. - Донецк: ДонНТУ, 2016. - №3 (5). – С. 65-71.

5. Паттерсон, Д. Архитектура компьютера и проектирование компьютерных систем. / Д. Паттерсон, Дж. Хеннесси. - 4-е изд. – Санкт-Петербург: Питер, 2012. – 784 с.

6. Таненбаум, Э. Архитектура компьютера. / Э. Таненбаум, Т. Остин. - 6-е изд. – Санкт-Петербург: Питер, 2014. – 816 с.

7. Авксентьева, О. А. Разработка специализированного устройства на базе ПЛИС для реализации операции сложения и вычитания чисел с плавающей запятой / О. А. Авксентьева, Д. И. Выростков, Р. В. Мальчева // Информатика и кибернетика. – Донецк.: ДонНТУ. – 2020. – № 4(22). - С.66-72.

8. Мальчева, Р. В. FPGA-реализация векторно-матричных умножений / Р. В. Мальчева, А. И. Воронова, И. И. Дегтярева // Материалы XII Международной научно-технической конференции «Информатика, управляющие системы, математическое и компьютерное моделирование» (ИУСМКМ – 2021). - Донецк: ДонНТУ, 2021. - С.145-148.

Николаенко Д.В., Выростков Я.И., Володько Л.П. Организация АЛУ с учетом архитектурных особенностей и оптимизаций. *В статье рассмотрены подходы к организации*

АЛУ с учетом его функциональных и архитектурных особенностей. Описаны основные виды операций, выполняемых в АЛУ, включая операции над двумя операндами и операции сдвига. Особое внимание уделено оптимизации структуры АЛУ: сравниваются варианты использования отдельных схем для каждой операции, универсальной схемы и гибридного подхода, при котором операции разделяются на группы. Рассматривается роль сдвигов в алгоритмах умножения и деления, а также предлагается подход с выделением отдельного модуля для выполнения всех операций сдвига. Обсуждаются компромиссы между быстродействием, сложностью схемы и экономией аппаратных ресурсов, приводящие к выбору оптимальной архитектуры.

Ключевые слова: *арифметико-логическое устройство, архитектура вычислительных устройств, оптимизация схемы, гибридный подход, быстродействие, упрощение архитектуры, аппаратные ресурсы.*

Nikolaenko Denis., Vyrostkov Yarosalv. ALU Organization Taking into Account Architectural Features and Optimizations. *The article considers approaches to organizing an arithmetic logic unit taking into account its functional and architectural features. The main types of operations performed in the ALU are described, including operations on two operands and shift operations. Particular attention is paid to optimizing the ALU structure: the options for using separate circuits for each operation, a universal circuit, and a hybrid approach in which operations are divided into groups are compared. The role of shifts in multiplication and division algorithms is considered, and an approach is proposed with the allocation of a separate module for performing all shift operations. Tradeoffs between performance, circuit complexity, and saving hardware resources are discussed, leading to the choice of the optimal architecture.*

Keywords: *arithmetic logic unit, computing device architecture, circuit optimization, hybrid approach, performance, architecture simplification, hardware resources.*