СРАВНИТЕЛЬНЫЙ АНАЛИЗ АРХИТЕКТУР ЛОГИЧЕСКИХ И АРИФМЕТИЧЕСКИХ БЛОКОВ МИКРОПРОЦЕССОРА

Халиуллина М.Н., Жуковский А.Д.

Национальный детский технопарк, г. Минск, Республика Беларусь

Биран С.А.

В работе выполнен сравнительный анализ архитектур арифметико-логических устройств (АЛУ) микропроцессоров. Рассмотрены принципы работы сумматоров, умножителей, устройств деления и логических блоков. Выявлены преимущества и недостатки сумматоров с ускоренным переносом, матричных сдвигателей, а также особенности реализации в CISC-, RISC- и VLIW-архитектурах. Определены ключевые ограничения быстродействия. Показано, что RISC-архитектуры обеспечивают более высокую энергоэффективность за счёт упрощения блоков, тогда как CISC ориентированы на выполнение сложных инструкций.

В современных процессорах ключевую роль играет АЛУ. Оно отвечает за непосредственную обработку данных, управляя вычислительными процессами с двоичными кодами на уровне отдельных битов и регистров.

Требования к быстродействию АЛУ стремительно растут. Обучение моделей искусственного интеллекта и работа с Big Data требуют большого количество энергозатрат, что делает оптимизацию архитектуры АЛУ актуальной научно-технической задачей. В большинстве АЛУ предусмотрена возможность выполнения основных арифметических и логических операций, а также сдвигов, что позволяет непосредственно управлять микропроцессором [1]. Целью данной работы является анализ различных подходов к проектированию блоков АЛУ и определение их эффективности для разных типов задач.

На рисунке 1 представлена схема того, как современные АЛУ используют одновременно несколько блоков исполнения одной операции путём сортировки данных.

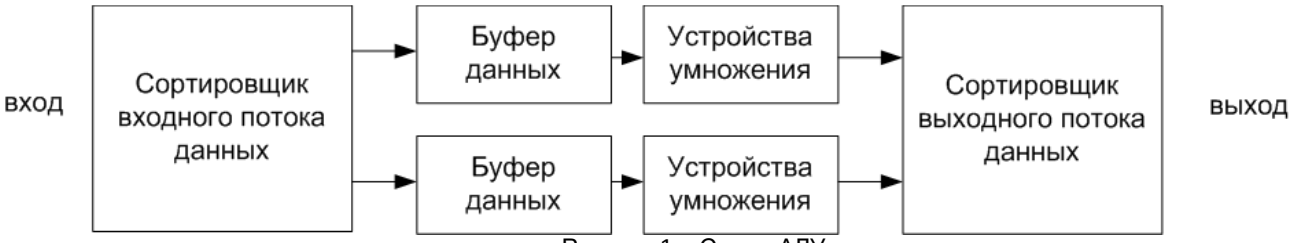

Рисунок 1 – Схема АЛУ

Основной операцией в АЛУ является сложение, так как через него реализуется вычитание, а подобные оптимизации используются для реализации умножения и деления.

В сумматорах наиболее сложным элементом для реализации является перенос разряда (англ. Carry Chain). В простых сумматорах (англ. Ripple Carry Adder) перенос идет последовательно от младшего бита к старшему, это делает процесс медленным. В современных архитектурах используют сумматоры с ускоренным переносом (англ. Carry-Lookahead), которые вычисляют перенос сразу для групп битов [2].

При проектировании умножителей для повышения эффективности часто используется пороговое разделение алгоритмов в зависимости от разрядности чисел. Порогом в данном контексте является максимальная разрядность, при которой аппаратная сложность прямого перемножения остается допустимой. Если входные числа имеют разрядность ниже пороговой, они перемножаются классическим матричным способом. Если же разрядность выше порога, каждое число разбивается на две части (старшие и младшие разряды), после чего из этих частей формируются промежуточные суммы. Затем вычисляется перекрёстное произведение, а полученный результат последовательно сдвигается для согласования разрядов. Это позволяет эффективно перемножать числа высокой разрядности, комбинируя операции с их фрагментами [3].

Для ускорения умножения применяются два основных метода: алгоритм Бута и деревья Уолесса. Алгоритм Бута позволяет сократить количество промежуточных сложений посредством анализа группы битов множителя (например, заменяя последовательность единиц одним вычитанием и одним сложением). При таком построении исключается распространение переноса от старших битов слагаемых к младшим. В связи с этим распространение переноса на этапе декодирования Бута вносит значительный вклад в суммарное время умножения [4]. Деревья Уоллеса – это способ быстрой расстановки сумматоров в виде дерева. Дерево преобразует 6 входных бит в 2 выходных бита – бит суммы и бит переноса [5]. Они позволяют складывать все промежуточные результаты умножения параллельно, сокращая время исполнения до логарифмического (время выполнения операции пропорционально логарифму от количества перемножаемых разрядов, O(log(n)), а не их количеству O(n)).

Блоки деления являются наиболее сложными и медленными. Обычно реализуются через итерационные алгоритмы, которые реализуются в несколько тактов.

Логические операции реализуются проще, так как между битами нет зависимостей (переносов).

Базовые операции выполняются набором логических вентилей через мультиплексор, который выбирает нужный результат на основе кода инструкции. Для быстрого сдвига числа на произвольное количество бит используется «матричный сдвигатель» (англ. Barrel Shifter). Он позволяет выполнить операцию за один такт, что критично для работы с числами с плавающей запятой и упаковки данных. По результатам операций логический блок формирует флаги, которые записываются в регистр состояния и используются для ветвления программ.

Архитектура АЛУ напрямуюзависит от процессора:

1 CISC (например, Intel/AMD) сложные блоки, ориентированные на выполнение тяжелых инструкций. Одно АЛУ может выполнять очень много функций, но это усложняет его схему и повышает энергопотребление.

2 RISC (например, ARM, Apple Silicone, RISC-V) из упрощенных блоков, то есть вместо одного сложного АЛУ часто используется несколько простых, работающих параллельно (суперскалярность). Это позволяет поднять частоту и снизить нагрев.

Основными критериями АЛУ являются производительность, энергопотребление, масштабируемость [6].

Таблица 1 –Сравнение архитектур

Архитектура	Набор инструкций	Реализация логических операций	Главное ограничение
CISC (x86, VAX)	Большой, переменной длины; многие операции составные	Логика реализуется в отдельном ALU- блоке; через традиционную цепь переноса	Вариативность задержек из-за микрокода,, медленный путь переноса
RISC (ARM, RISC-V, MIPS)	Фиксированная длина (обычно 32 бит), набор простых инструкций	Логика использует сумматор с ускоренным переносом и сдвигатель.	Цепь переноса добавляет один уровень логики; сдвиги требуют мультиплексоры (2-3 такта в худшем случае)
EPIC/ VLIW (Itanium, Intel Xeon)	Пакет из 4–8 параллельных под-инструкций фиксированной длины (128–256 бит)	Каждый слот использует обычный RISC-ALU	Не устраняет задержку цепи переноса и мультиплексоры; эффективность сильно зависит от качества компилятор-раскладки

В ходе работы выполнен сравнительный анализ архитектур АЛУ. Рассмотрены операции АЛУ, такие как сложение с ускоренным переносом, умножения, алгоритм Бута и деревья Уоллеса, блоки деления. Сравнение архитектур показало, что CISC ориентированы на выполнение сложных инструкций ценой усложнения схемы и повышенного энергопотребления, тогда как RISC используют несколько простых параллельных блоков, что повышает частоту и снижает энергоэффективность. Эффективность VLIW сильно зависит от качества компиляторной раскладки (статический процесс планирования параллельного выполнения инструкций, осуществляемый компилятором на этапе компиляции, а не аппаратными средствами процессора во время исполнения). Выбор архитектуры АЛУ определяется требуемой производительностью, энергопотреблением и масштабируемостью.

Список использованных источников:

1. Богомолов, Б. К. Проектирование АЛУ на языке VHDL в программе Aldec Active-HDL / Б. К Богомолов, Н. А. Юрченко // Современные проблемы телекоммуникаций. Материалы Всероссийской научно-технической конференции. – 2024. – С. 265–269.

2. Федюнин, Р. Н. Временной анализ и реализация аппаратно-программных модулей арифметического логического устройства / Р. Н. Федюнин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2016. – № 2 (38). – С. 33–48.

3. Федюнин, Р. Н. Блок арифметико-логического устройства для реализации умножения больших чисел / Р. Н Федюнин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2016. – № 3 (39). – С. 29-40.

4. Гармаш, А. А. Сравнительный анализ простых матричных умножителей и умножителей, реализованных по алгоритму Бута / А. А. Гармаш // Московский инженерно-физический институт (государственный университет).

5. Паулин О. Н. Об ускорении свертки многорядных кодов / О. Н. Паулин // Одесский национальный политехнический университет. – 2013.

6. Таненбаум, Э. Э. Архитектура компьютера / Э. Таненбаум, Т. Остин. – 6-е изд. – СПб.: Питер, 2013. – 816 с.