Основы микропроцессорной техники

1.Философия микропроцессорной техники: В этой лекции рассказывается о базовой терминологии микропроцессорной техники, о принципах организации микропроцессорных систем, о структуре связей, режимах работы и об основных типах микропроцессорных систем.

В этой главе рассматриваются базовые концепции, которые лежат в основе любой *микропроцессорной системы* — от простейшего микроконтроллера до сложного компьютера. Именно в этом смысле здесь используется термин «философия».

Для начала несколько основных определений.

- **Электронная система** — в данном случае это любой электронный узел, блок, прибор или комплекс, производящий обработку информации.
- **Задача** — это набор функций, выполнение которых требуется от электронной системы.
- **Быстродействие** — это показатель скорости выполнения электронной системой ее функций.
- **Гибкость** — это способность системы подстраиваться под различные задачи.
- **Избыточность** — это показатель степени соответствия возможностей системы решаемой данной системой задаче.
- **Интерфейс** — соглашение об обмене информацией, правила обмена информацией, подразумевающие электрическую, логическую и конструктивную совместимость устройств, участвующих в обмене. Другое название — **сопряжение**.

Микропроцессорная система может рассматриваться как частный случай электронной системы, предназначенной для обработки входных сигналов и выдачи выходных сигналов (рис. 1.1). В качестве входных и выходных сигналов при этом могут использоваться аналоговые сигналы, одиночные цифровые сигналы, цифровые коды, последовательности цифровых кодов. Внутри системы может производиться хранение, накопление сигналов (или информации), но суть от этого не меняется. Если система цифровая (а *микропроцессорные системы* относятся к разряду цифровых), то входные аналоговые сигналы преобразуются в последовательности кодов выборок с помощью АЦП, а выходные аналоговые сигналы формируются из последовательности кодов выборок с помощью ЦАП. Обработка и хранение информации производятся в цифровом виде.

Характерная особенность традиционной цифровой системы состоит в том, что алгоритмы обработки и хранения информации в ней жестко связаны со схемотехникой системы. То есть изменение этих алгоритмов возможно только путем изменения структуры системы, замены электронных узлов, входящих в систему, и/или связей между ними. Например, если нам нужна дополнительная операция суммирования, то необходимо добавить в структуру системы лишний сумматор. Или если нужна дополнительная функция хранения кода в течение одного такта, то мы должны добавить в структуру еще один регистр. Естественно, это практически невозможно сделать в процессе эксплуатации, обязательно нужен новый производственный цикл проектирования, изготовления, отладки всей системы. Именно поэтому традиционная цифровая система часто называется системой на «жесткой логике».

Любая система на «жесткой логике» обязательно представляет собой специализированную систему, настроенную исключительно на одну задачу или (реже) на несколько близких, заранее известных задач. Это имеет свои бесспорные преимущества.

Во-первых, специализированная система (в отличие от универсальной) никогда не имеет аппаратурной избыточности, то есть каждый ее элемент обязательно работает в полную силу (конечно, если эта система грамотно спроектирована).

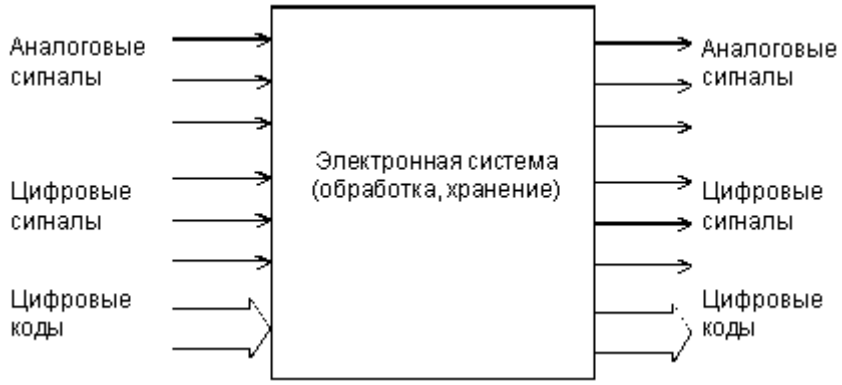

Рис. 1.1. Электронная система.

Во-вторых, именно специализированная система может обеспечить максимально высокое быстродействие, так как скорость выполнения алгоритмов обработки информации определяется в ней только быстродействием отдельных логических элементов и выбранной схемой путей прохождения информации. А именно логические элементы всегда обладают максимальным на данный момент быстродействием.

Но в то же время большим недостатком цифровой системы на «жесткой логике» является то, что для каждой новой задачи ее надо проектировать и изготавливать заново. Это процесс длительный, дорогостоящий, требующий высокой квалификации исполнителей. А если решаемая задача вдруг изменяется, то вся аппаратура должна быть полностью заменена. В нашем быстро меняющемся мире это довольно расточительно.

Путь преодоления этого недостатка довольно очевиден: надо построить такую систему, которая могла бы легко адаптироваться под любую задачу, перестраиваться с одного алгоритма работы на другой без изменения аппаратуры. И задавать тот или иной алгоритм мы тогда могли бы путем ввода в систему некой дополнительной управляющей информации, **программы** работы системы (рис. 1.2). Тогда система станет универсальной, или **программируемой**, не жесткой, а гибкой. Именно это и обеспечивает *микропроцессорная система*.

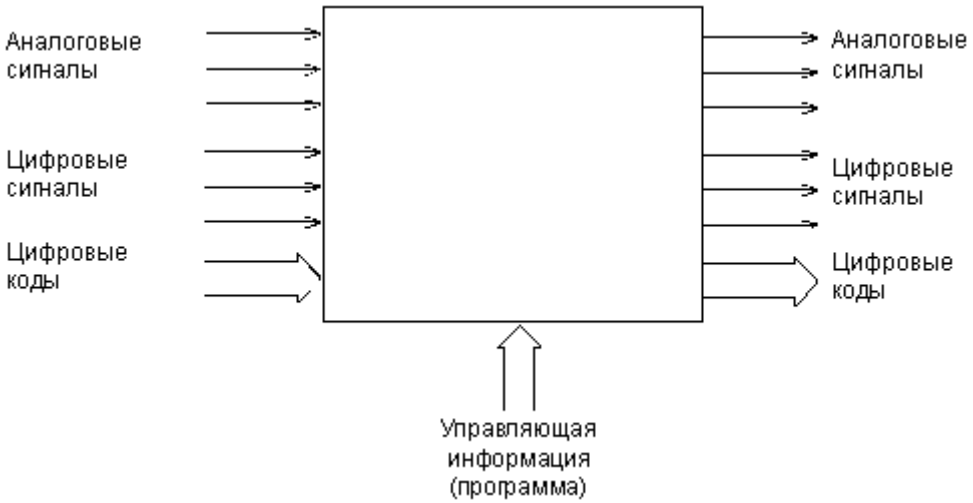

Рис. 1.2. Программируемая (она же универсальная) электронная система.

Но любая универсальность обязательно приводит к избыточности. Ведь решение максимально трудной задачи требует гораздо больше средств, чем решение максимально простой задачи. Поэтому сложность универсальной системы должна быть такой, чтобы обеспечивать решение самой трудной задачи, а при решении простой задачи система будет работать далеко не в полную силу, будет использовать не все свои ресурсы. И чем проще решаемая задача, тем больше избыточность, и тем менее оправданной становится универсальность. Избыточность ведет к увеличению стоимости системы, снижению ее надежности, увеличению потребляемой мощности и т.д.

Кроме того, универсальность, как правило, приводит к существенному снижению быстродействия. Оптимизировать универсальную систему так, чтобы каждая новая задача

решалась максимально быстро, попросту невозможно. Общее правило таково: чем больше универсальность, гибкость, тем меньше быстродействие. Более того, для универсальных систем не существует таких задач (пусть даже и самых простых), которые бы они решали с максимально возможным быстродействием. За все приходится платить.

Таким образом, можно сделать следующий вывод. Системы на «жесткой логике» хороши там, где решаемая задача не меняется длительное время, где требуется самое высокое быстродействие, где алгоритмы обработки информации предельно просты. А универсальные, программируемые системы хороши там, где часто меняются решаемые задачи, где высокое быстродействие не слишком важно, где алгоритмы обработки информации сложные. То есть любая система хороша на своем месте.

Однако за последние десятилетия быстродействие универсальных (*микропроцессорных*) систем сильно выросло (на несколько порядков). К тому же большой объем выпуска микросхем для этих систем привел к резкому снижению их стоимости. В результате область применения систем на «жесткой логике» резко сузилась. Более того, высокими темпами развиваются сейчас программируемые системы, предназначенные для решения одной задачи или нескольких близких задач. Они удачно совмещают в себе как достоинства систем на «жесткой логике», так и программируемых систем, обеспечивая сочетание достаточно высокого быстродействия и необходимой гибкости. Так что вытеснение «жесткой логики» продолжается.

1.1. Что такое микропроцессор?

Ядром любой *микропроцессорной системы* является *микропроцессор* или просто процессор (от английского processor). Перевести на русский язык это слово правильнее всего как «обработчик», так как именно ***микропроцессор*** — это тот узел, блок, который производит всю обработку информации внутри *микропроцессорной системы*. Остальные узлы выполняют всего лишь вспомогательные функции: хранение информации (в том числе и управляющей информации, то есть программы), связи с внешними устройствами, связи с пользователем и т.д. Процессор заменяет практически всю «жесткую логику», которая понадобилась бы в случае традиционной цифровой системы. Он выполняет арифметические функции (сложение, умножение и т.д.), логические функции (сдвиг, сравнение, маскирование кодов и т.д.), временное хранение кодов (во внутренних регистрах), пересылку кодов между узлами *микропроцессорной системы* и многое другое. Количество таких элементарных операций, выполняемых процессором, может достигать нескольких сотен. Процессор можно сравнить с мозгом системы.

Но при этом надо учитывать, что все свои операции процессор выполняет **последовательно**, то есть одну за другой, по очереди. Конечно, существуют процессоры с параллельным выполнением некоторых операций, встречаются также *микропроцессорные системы*, в которых несколько процессоров работают над одной задачей параллельно, но это редкие исключения. С одной стороны, последовательное выполнение операций — несомненное достоинство, так как позволяет с помощью всего одного процессора выполнять любые, самые сложные алгоритмы обработки информации. Но, с другой стороны, последовательное выполнение операций приводит к тому, что время выполнения алгоритма зависит от его сложности. Простые алгоритмы выполняются быстрее сложных. То есть *микропроцессорная система* способна сделать все, но работает она не слишком быстро, ведь все информационные потоки приходится пропускать через один-единственный узел — *микропроцессор* (рис. 1.3). В традиционной цифровой системе можно легко организовать параллельную обработку всех потоков информации, правда, ценой усложнения схемы.

Итак, *микропроцессор* способен выполнять множество операций. Но откуда он узнает, какую операцию ему надо выполнять в данный момент? Именно это определяется управляющей информацией, программой. Программа представляет собой набор **команд (инструкций)**, то есть цифровых кодов, расшифровав которые, процессор узнает, что ему надо делать. Программа от начала и до конца составляется человеком, программистом, а процессор выступает в роли послушного исполнителя этой программы, никакой инициативы он не проявляет (если, конечно, исправен). Поэтому сравнение процессора с мозгом не слишком корректно. Он всего лишь исполнитель того алгоритма, который заранее составил для него человек. Любое отклонение от этого алгоритма может быть вызвано только неисправностью процессора или каких-нибудь других узлов *микропроцессорной системы*.

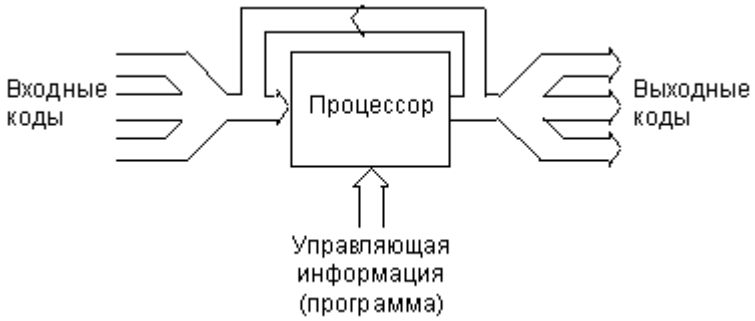

Рис. 1.3. Информационные потоки в микропроцессорной системе.

Все команды, выполняемые процессором, образуют **систему команд** процессора. Структура и объем системы команд процессора определяют его быстродействие, гибкость, удобство использования. Всего команд у процессора может быть от нескольких десятков до нескольких сотен. Система команд может быть рассчитана на узкий круг решаемых задач (у специализированных процессоров) или на максимально широкий круг задач (у универсальных процессоров). Коды команд могут иметь различное количество разрядов (занимать от одного до нескольких байт). Каждая команда имеет свое время выполнения, поэтому время выполнения всей программы зависит не только от количества команд в программе, но и от того, какие именно команды используются.

Для выполнения команд в структуру процессора входят внутренние регистры, арифметико-логическое устройство (АЛУ, ALU — Arithmetic Logic Unit) , мультиплексоры, буферы, регистры и другие узлы. Работа всех узлов синхронизируется общим внешним тактовым сигналом процессора. То есть процессор представляет собой довольно сложное цифровое устройство (рис. 1.4).

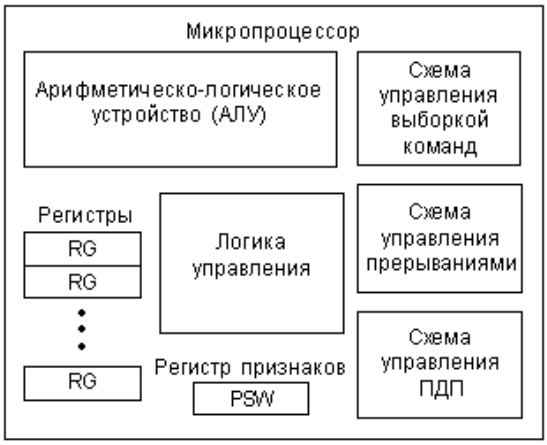

Рис. 1.4. Пример структуры простейшего процессора.

Впрочем, для разработчика *микропроцессорных систем* информация о тонкостях внутренней структуры процессора не слишком важна. Разработчик должен рассматривать процессор как «черный ящик», который в ответ на входные и управляющие коды производит ту или иную операцию и выдает выходные сигналы. Разработчику необходимо знать систему команд, режимы работы процессора, а также правила взаимодействия процессора с внешним миром или, как их еще называют, протоколы обмена информацией. О внутренней структуре процессора надо знать только то, что необходимо для выбора той или иной команды, того или иного режима работы.