6.1 Основные узлы процессора

В предыдущих лекциях мы рассмотрели организацию компьютера на традиционном машинном уровне. Мы знаем как представлены данные, знаем структуры и форматы машинных команд, представляем обобщенную структуру процессора. Теперь рассмотрим логику его работы (рис. 6.1.1).

И так, как мы уже знаем основным узлом процессора является Арифметико-логическое устройство (АЛУ).

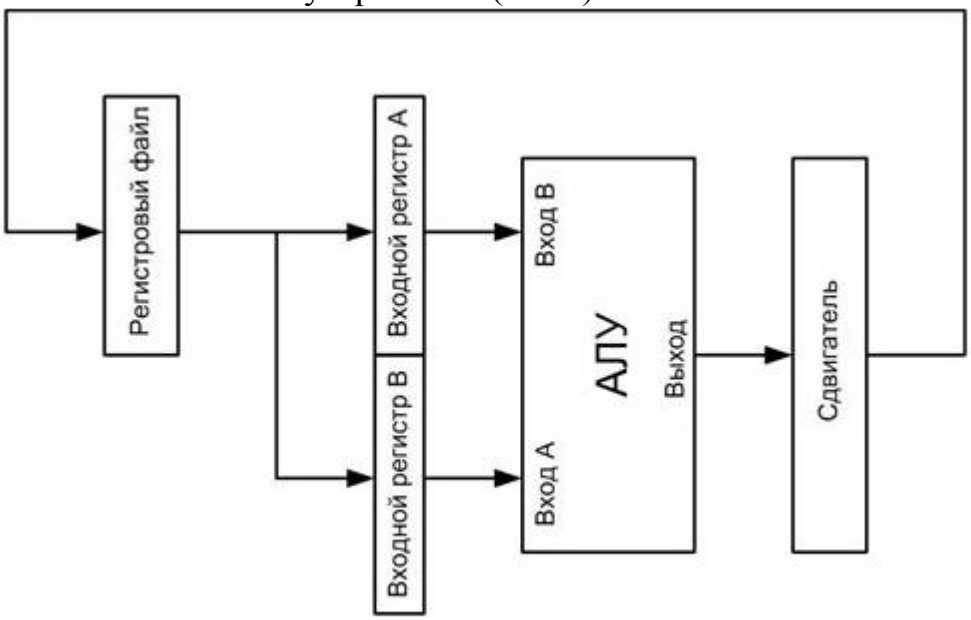

Рис. 6.1.1 Простейший тракт данных

АЛУ представляет собой комбинационную схему, состояние выхода которой соответствует какой-либо логической или арифметической функции от входных данных. Когда данные на входе есть – на выходе есть результат, когда нет – нет и результата. Соответственно, пока АЛУ выполняет операцию над данными, они должны где-то храниться. Например, **в регистре**. Для выполнения большинства операции, АЛУ необходимо два аргумента и для них в нем имеется два входа. Соответственно, и регистров для входных данных АЛУ должно быть два.

Устройство для сдвига данных (сдвигатель). Для организации вычислений и обработки данных очень часто бывает необходимо сдвигать слово данных на заданное число разрядов в ту или иную сторону. При помощи АЛУ можно сдвинуть слово данных в сторону старшего разряда, если сложить число с самим собой. Однако необходим сдвиг и в обратную сторону, а при помощи АЛУ его выполнить нельзя. Также может быть необходим сдвиг данных и на большее число разрядов за один раз. Поэтому имеет прямой смысл объединить АЛУ с параллельным сдвигателем в один блок, и получать результат операции не с выхода АЛУ, а с выхода сдвигателя, расположенного следом за ним.

Регистровый файл. Когда мы получим результат операции над двумя аргументами, хранящимися во входных регистрах, прежде чем переходить к следующей операции необходимо этот результат где-то сохранить. Вообще

говоря, данные можно сохранять и в оперативной памяти, внешней по отношению к процессору. Однако обращение к внешней памяти представляет собой не самое простое действие, и занимает некоторое, сравнительно большое время. Поэтому внутри процессора почти всегда делают регистровый файл, состоящий из нескольких регистров. Причем чем больше регистров будет внутри процессора, тем лучше.

Основной тракт данных. В регистровом файле должен сохраняться результат выполненной операции. Но при этом, для выполнения операции необходимо извлекать данные, хранящиеся в регистровом файле для использования в качестве аргументов для операции. Получается, что выход сдвигателя должен быть соединен с входом регистрового файла, а выход регистрового файла с входом входных регистров АЛУ. Таким образом, мы пришли туда, откуда начали, т.е. у нас образовалось кольцо из узлов обработки и хранения данных. Это кольцо мы будем называть **основным трактом данных** (рис. 6.1.1).

Помимо перечисленных узлов необходимы и другие, вспомогательные узлы и необходима возможность передавать данные в эти узлы из тракта данных, и наоборот, из этих узлов в тракт данных:

1. Необходим **счетчик команд** для хранения адреса текущей команды, с возможностью автоматического увеличения, а также считывания и записи данных.
2. Необходим специальный **регистр состояния** для хранения признаков результата выполнения операции, таких как равенство операндов, перенос при сложении, перенос при сдвиге.

Кроме того, необходимо иметь возможность приема и передачи данных в оперативную память и внешние устройства. Обычно для этого используют **внешнюю шину**. Для организации взаимодействия процессора с внешней шиной, необходимо иметь следующие возможности:

3. Принимать данные с внешней шины, и сохранять их, для дальнейшей передачи другим узлам – необходим **регистр входных данных шины**.
4. Принимать данные от других узлов процессора, для передачи на шину данных – необходим **регистр выходных данных шины**.
5. Принимать данные от других узлов процессора, для передачи на шину адреса при обращении к аргументам команд, хранящимся в оперативной памяти – необходим **регистр адреса шины**.
6. Передавать на шину адреса содержимое регистра адреса шины при обращении к памяти в процессе выполнения команды и содержимое счетчика команд при выборке самой команды – необходим **селектор адреса шины**.

Ну и наконец, необходим регистр для хранения самой команды в процессе выполнения. Этот регистр должен принимать данные с шины и передавать их управляющим схемам.

Часть содержимого регистра команд – адреса регистров, в которых хранятся операнды, должна коммутироваться и передаваться в регистровый файл. Для этого необходимо специальное **коммутационное устройство**.

Два способа коммутации данных в процессоре Передача данных во все эти узлы не вызывает особых проблем, поскольку все их входы, предназначенные для загрузки в них данных, можно подключить к любому участку основного тракта данных, например выходу сдвигателя. А вот передача информации из этих узлов в тракт данных процессора представляет некоторую сложность из-за невозможности соединения выходов. Решить эту проблему можно одним из двух способов. Первый способ состоит в том, чтобы оснастить выход сдвигателя, а также выходы всех этих узлов буферными схемами с тремя состояниями выхода и синхронизировать их работу так, чтобы в нужный момент времени только один узел мог передавать данные. Тогда образуется внутренняя шина данных процессора. Другой способ состоит в том, чтобы на каком-либо из участков тракта данных, разместить дополнительный селектор с несколькими входами, к каждому из которых подключен один из этих дополнительных узлов. Выбор нужного входа в этом случае позволит выбирать, из какого именно узла мы хотим получить данные.

6.2 Микрокоманды и микропрограммы

Задача управления трактом данных процессора состоит в формировании различных последовательностей управляющих сигналов в зависимости от кода операции выполняемой команды, т.е. старшего байта регистра команд, и сигналов о состоянии процессора и шины. В первую очередь для этого необходимо ввести квантование времени, т.е. разделить время на равные промежутки. Для этого используется тактовый генератор, вырабатывающий тактовый сигнал. Тогда для каждого периода тактового сигнала необходимо сформировать слово, состоящее из нужного числа разрядов (по числу управляющих сигналов тракта данных) и использовать разряды этого слова в качестве управляющих сигналов. Такое слово называется микрокомандой. Каждой команде должна соответствовать последовательность микрокоманд – микропрограмма. Каждая микропрограмма должна заканчиваться выборкой из оперативной памяти следующей команды и записью ее в регистр команд. После этого начнется выполнение следующей микропрограммы, и так до бесконечности.

Этапы выполнения микрокоманды. Каждая микрокоманда представляет собой управляющее слово, которое должно где-то храниться. Ее извлечение потребует какого-то времени, а ее выполнение можно начинать только после ее извлечения. Если микрокоманда должна выполняться за один период тактового сигнала, то тактовый сигнал должен быть поделен на еще более мелкие промежутки времени. Тогда на первом промежутке будет происходить извлечение микрокоманды, а на втором – ее выполнение. Причем ясно, что выполнение микрокоманды будет

происходить дольше, чем ее извлечение, поскольку потребует выполнения нескольких действий – извлечения аргументов, выполнения операции и записи результата.

Этапы выполнения микрокоманды могут совмещаться. Поскольку в приведенной блок-схеме (рис. 6.1.1) на входе АЛУ предусмотрены буферные регистры, выполнение микрокоманды может быть разбито на два этапа. На первом этапе будет происходить извлечение аргументов из регистрового файла и запись в промежуточные регистры, а на втором – выполнение операции и запись результатов в регистровый файл. Для выполнения большинства микрокоманд необходимы аргументы, хранящиеся в регистровом файле по адресам, указанным в качестве аргументов A и B в выполняемой команде. Для извлечения этих аргументов самой микрокоманды не требуется, и поэтому извлечение аргументов можно совместить по времени с извлечением самой микрокоманды. Тогда выполнение одной микрокоманды будет состоять из двух этапов. На первом этапе будет происходить извлечение микрокоманды, одновременно с извлечением аргументов, а на втором – выполнение микрокоманды и запись результата.

Временные метки. Если каждая микрокоманда должна выполняться за один период тактового сигнала, то этот период должен быть разбит на более мелкие промежутки времени. Если разбить период тактового сигнала пополам, то в первой половине периода нужно извлечь микрокоманду и аргументы, а на втором - выполнить микрокоманду и записать результат. Аргументы, извлеченные на первом этапе, должны быть зафиксированы во входных регистрах АЛУ строго в середине периода тактового сигнала. Результат выполнения операции должен быть зафиксирован в регистровом файле или в других узлах строго в конце периода тактового сигнала. Для этого необходимы временные метки двух типов – в середине и в конце периода тактового сигнала. Если считать тактовый сигнал симметричным сигналом прямоугольной формы, то временные метки могут представлять собой момент изменения уровня тактового сигнала, т.е. его положительный и отрицательный фронты.

Типы узлов процессора. Узлы, из которых состоит процессор, относятся к двум основным типам – узлы комбинационного типа и узлы с памятью. Для узлов комбинационного типа состояние их выходов определяется только состоянием входных сигналов в данный момент времени. Для узлов с памятью состояние выходов определяется состоянием входных сигналов в момент изменения специального управляющего сигнала и не меняется в другие моменты времени. Если для работы узлов комбинационного типа достаточно, чтобы управляющие сигналы оставались стабильными до момента фиксации результата в следующем за ним узле с памятью, то для узлов с памятью необходимы управляющие сигналы, начало и конец которых, строго соответствуют временным меткам. Например, для правильной работы АЛУ вполне достаточно, чтобы управляющие сигналы оставались стабильными от момента появления входных данных до момента

фиксации результата, а их поведение вне этого временного промежутка значения не имеет. Тогда как для входного регистра АЛУ важно, чтобы данные в нем были зафиксированы строго в определенный момент времени, а не раньше и не позже.

Для простоты можно договориться применять в схеме процессора только такие узлы с памятью, фиксация данных в которых осуществляется в момент изменения управляющего сигнала от нуля к единице.

6.3 Процессор как совокупность автоматов

Для более ясного понимания принципов работы процессора будем рассматривать процессор как набор автоматов.

Любой процессор обычно содержит три основные части: устройство обработки данных, состоящее из набора операционных устройств иначе АЛУ, устройство управления (УУ) и блок интерфейса (БИ) для сопряжения с системной магистралью (рис. 6.3).

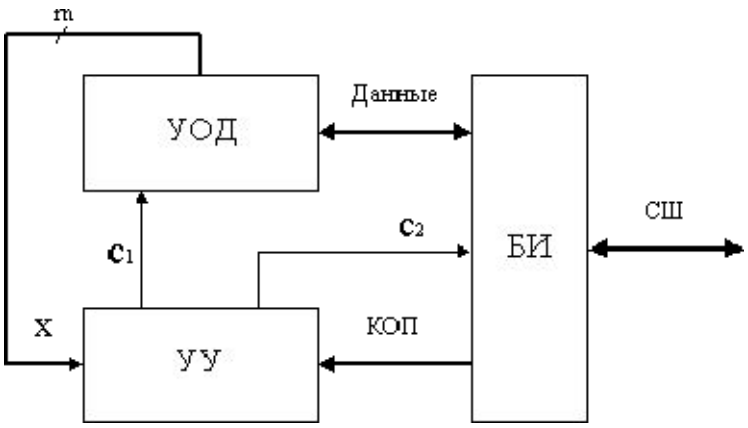

Рис. 6.3.1 Структура процессора

Устройство обработки данных процессора АЛУ-представляет собой операционный автомат, реализующий алгоритмы выполнения арифметических и логических операций.

Устройство управления процессора представляет собой *управляющий автомат*, предназначенный для выработки в каждом такте работы процессора векторов управляющих сигналов: с1 – для управления АЛУ и с2 – для управления устройствами БИ. Входной информацией для выработки управляющих сигналов является код операции (КОП), выполняемой процессором, и набор сигналов X, информирующий УУ о состоянии АЛУ и результатах выполнения предыдущих микроопераций.

Блок интерфейса процессора представляет собой автомат, выполняющий функции операционного и управляющего автоматов. Он обеспечивает взаимодействие процессора со всеми внешними устройствами. В их число входят: узел управления шинами – **управляющий автомат**; узел формирования адресов памяти– **операционный автомат**; буферные схемы и некоторые другие вспомогательные узлы – выполняющие функции электрического согласования.

Операционный автомат – устройство, объединяющее все функциональные модули, непосредственно занятые обработкой информации, ее хранением, пересылкой и т.п. У этих модулей имеются *информационные* входы и выходы, *управляющие* входы, также выходы, которые сигнализируют о состоянии вычислительного процесса (код знака операнда, признак переноса, признак обнуления результата и др.). Эти выходы мы будем называть *осведомительными сигналами,* или *внутренними логическими условиями* устройства.

Осведомительные сигналы - выходные сигналы операционного автомата, информирующие о состоянии вычислительного процесса (код знака операнда, признак переноса, признак обнуления результата и др.)

Управляющий автомат это устройство, вырабатывающее сигналы управления по заданной программе и с учетом значений внутренних и внешних логических условий, которые для него являются входными переменными. Внешние логические условия задают одну из нескольких возможных в данном устройстве микропрограмм.

Управляющий автомат (УА) можно строить двумя способами.

Управляющий автомат с жесткой логикой – специальная логическая сеть с элементами памяти, реализующая микропрограмму выполнения команды. Структура сети определяется теми микропрограммами, которые должны выполняться данным устройством. При этом способе можно свести к минимуму объем оборудования и, что бывает гораздо важнее, обеспечить максимально возможное быстродействие. Зато проектирование такого управления является делом долгим и сложным, а перепрограммирование УА на другую работу вообще невозможно.

Микропрограммный УА, или **управляющий автомат с программируемой логикой**, строится с использованием специальной встроенной памяти микропрограмм. В эту память записываются все требуемые микропрограммы, а от схемы управления требуется только организация чтения микропрограмм. Проектирование микропрограммного автомата несравненно проще, так как в нем используются типовые аппаратные модули. Единственной специальной задачей, решаемой в рамках теории автоматов, остается абстрактный синтез и составление таблиц программирования памяти. Недостатками микропрограммного управления является некоторая избыточность оборудования и меньшее быстродействие.

Определим некоторые общие понятия и функциональное назначение устройства управления. Определим, прежде всего, понятие микрокоманды, а также взаимосвязанных с ним понятий микрооперации и микропрограммы.

Программа представляет собой совокупность команд, записанных в определенной последовательности, которая обеспечивает решение данной конкретной задачи на компьютере.

Команда - это инструкция для выполнения очередного этапа в вычислениях, а также соответствующее обозначение этой инструкции.

Операция - действие, выполняемое в компьютере или процессоре под воздействием команды. При этом каждой команде соответствует операция, выполняемая процессором.

Практически в любом процессоре операция не является элементарным действием. Она состоит из последовательности нескольких других элементарных действий, которые называют *микрооперациями*. При этом одна или несколько совместимых во времени микроопераций выполняются за один элементарный интервал времени, представляющий собой период синхронизирующих (тактовых) импульсов и называемый *тактом*.

Таким образом, операция, выполняемая в процессоре под воздействием какой-либо команды, представляет собой ряд микроопераций. Каждая из микроопераций или несколько из них, выполняемых в один такт, реализуется в устройствах компьютера или процессора под воздействием *микрокоманды*. Следовательно, каждой команде соответствует своя совокупность микрокоманд. Эту совокупность микрокоманд, или микроинструкций, реализующих данную команду, называют *микропрограммой*.

Из изложенного видно, что в каждом такте в любом процессоре должна быть сформирована своя микрокоманда, или микроинструкция, которая и обеспечивает выполнение необходимых микроопераций. На рис. 6.3.2 представлена диаграмма, на которой по горизонтальной оси отображена последовательность тактов, а по вертикальной - номера цепей управления процессора при выполнении им некой команды. Каждому такту и каждой цепи управления поставлен в соответствие определенный уровень управляющего сигнала. Например, единичный уровень сигнала является разрешающим, а нулевой - запрещающим. Следует заметить, что для некоторых цепей за разрешающий может быть принят и нулевой уровень.

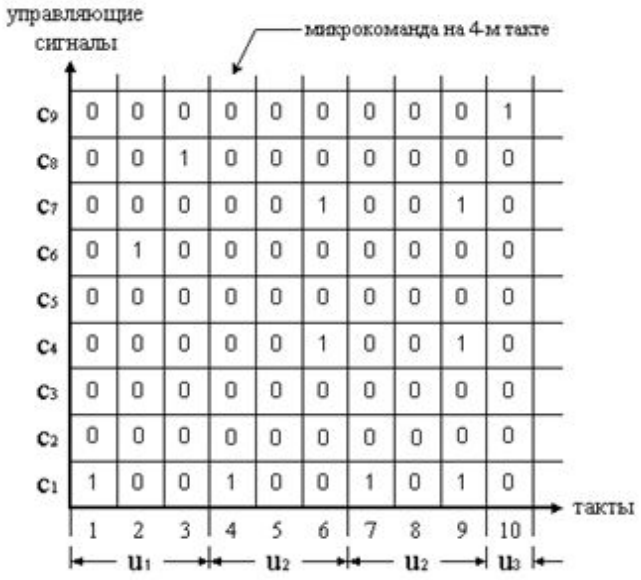

Рис. 6.3.2. Диаграмма распределения управляющих сигналов процессора по цепям управления и по тактам: *u*1,*u*2,*u*3 – коды команд

Из диаграммы видно, что в каждом такте на управляющие цепи процессора (общее число их может составлять 40 - 60 и более) должна быть

подана своя совокупность сигналов управления, разрешающих или запрещающих какую-то микрооперацию. Эту совокупность сигналов управления и называют *микрокомандой,* или *управляющим словом*. Считают, что в тех тактах, в которых имеются несколько разрешающих сигналов, одновременно выполняется несколько элементарных действий – микроопераций. Но при этом в каждом такте имеется только одно управляющее слово, т.е. реализуется одна микрокоманда.

Таким образом, при выполнении любой команды в процессоре должна быть сформирована совокупность управляющих сигналов, распределенных в пространстве (по цепям управления) и во времени (по тактам). Эта совокупность управляющих сигналов и представляет собой микропрограмму, реализующую данную команду.

Формирование управляющих сигналов для всех цепей управления в каждом такте осуществляется устройством управления процессора. Как видим, задача формирования управляющих сигналов является достаточно сложной и громоздкой, если принять во внимание при этом, что в процессоре имеется достаточно большое число цепей управления и значительное число команд (до 130 и более), причем каждая команда выполняется за несколько тактов (до 5 – 10 и более). Кроме того, устройство управления должно реагировать также на внешние сигналы управления.

В настоящее время для построения устройства управления в процессорах используются два принципа: 1) на основе аппаратной реализации или «жесткой» логики управления и 2) на основе микропрограммной реализации или «гибкой» логики управления. Рассмотрим кратко оба этих принципа. Но вначале необходимо рассмотреть некоторые общие узлы, которые используются при реализации обоих принципов построения устройства управления.

Во многих современных процессорах есть узел, который называют *очередью команд*. Очередь команд представляет собой запоминающее устройство, предназначенное для хранения очередных, подлежащих выполнению команд.

Заполнение буферной памяти команд производится в интервалы времени, когда шина данных процессора не занята им для обмена данными с основной памятью или внешними устройствами. Выборка команд из очереди команд производится процессором по мере их выполнения. Таким образом обеспечивается повышение общей производительности процессора, поскольку практически не требуется дополнительных затрат времени на выборку команд из оперативного запоминающего устройства - очередные команды уже находятся во внутренней буферной памяти процессора. Полное время на выборку команды с обращением к оперативному запоминающему устройству тратится лишь в тех случаях, когда производится передача управления при реализации условных переходов в программе. В этих случаях производится реинициализация очереди команд и загрузка ее новой последовательностью команд. При этом первая же выбранная из ОЗУ команда становится сразу доступной для выполнения. Одной из

разновидностей реализации очереди команд является так называемая КЭШ-память, размещаемая непосредственно на кристалле процессора. В ряде современных процессоров КЭШ-память используется при этом как буферное запоминающее устройство не только для потока команд, но также и для потока данных. Емкость такого буферного запоминающего устройства достигает 256 байт и более.

Устройство управления с программируемой логикой

Упрощенную структурную схему устройства управления с программируемой логикой можно представить в виде рис. 6.3.3. Двумя главными узлами этого устройства являются узел формирования адреса микрокоманды (NMIAF - Next MicroInstruction Address Former) и постоянное запоминающее устройство микрокоманд ROM(MI). Основной принцип действия такого устройства управления состоит в следующем. Вся совокупность микрокоманд, или управляющих слов, необходимых для реализации всего списка команд процессора на каждом такте их выполнения, хранится в постоянном запоминающем устройстве микрокоманд (хранилище микрокоманд).

Рис. 6.3.3. Микропрограммное устройство управления

При выполнении любой команды в каждом такте просто извлекаются из ПЗУ уже готовые для использования очередные микрокоманды. Каждая микрокоманда, или управляющее слово, имеет в ПЗУ свой адрес. Таким образом, чтобы выбрать микрокоманду, необходимую в очередном такте выполняемой команды, требуется сформировать ее адрес. Эту функцию

выполняет узел формирования адреса NMIAF. Исходными данными для формирования адреса помимо кода операции, логических условий из операционного устройства и сигналов о прерываниях является также информация, заключенная в адресной части предшествующей микрокоманды. При этом адресная часть предшествующей микрокоманды содержит в себе не только указания о том, как сформировать адрес очередной микрокоманды, но в необходимых случаях и сам адрес, если по условиям выполнения команды необходимо совершить переход в новую область ПЗУ.

Помимо основных узлов - ROM(MI) и NMIAF на структурной схеме (см. рис. 6.3.3) указан также ряд регистров: MIARG - регистр адреса микрокоманды, OPMIRG - регистр операционной части микрокоманды, AMIRG - регистр адресной части микрокоманды, которые выполняют вспомогательные функции временного хранения информации.

Микропрограммное устройство управления, как правило, реализуется в виде отдельного блока по отношению к собственно процессору и конструктивно выполняется на одной или нескольких отдельных микросхемах. Это дает возможность выбрать для реализации ПЗУ микроконтролера различные типы микросхем, различающихся как по организации, так и по принципу программирования. Среди имеющихся стандартных микросхем ПЗУ для рассматриваемого устройства управления могут быть выбраны:

- по организации – ПЗУ с произвольной выборкой либо программируемые логические матрицы;

- по принципу программирования – с однократным (ППЗУ) или многократным (РПЗУ) программированием.

Наиболее удобным для пользователя является вариант, основанный на применении микросхем ПЗУ с произвольной выборкой и возможностью однократного или многократного программирования.

Проведенное краткое рассмотрение микропрограммного устройства управления позволяет сделать некоторые обобщающие заключения. Как следует из анализа структурной схемы и принципа действия, данное устройство управления является достаточно сложным по своей аппаратурной реализации и требует значительного количества дополнительных микросхем. Кроме того, микропрограммирование такого устройства управления, т.е. программирование ПЗУ микрокоманд, является достаточно трудоемким процессом и требует большого внимания и аккуратности. Вместе с тем нельзя не отметить весьма широкие функциональные возможности микропрограммного устройства управления, которые могут оказаться особенно ценными при разработке специализированных компьютерных систем.

Общие особенности устройств управления с "гибкой" логикой:

- широкий и практически неограниченный набор команд; внутреннее программирование (на уровне микропрограмм), доступное пользователю, что создает возможность эмуляции набора команд любой известной системы или

разработки новой системы команд, наилучшим образом удовлетворяющей требованиям решения конкретной технической задачи;

- увеличение числа используемых микросхем, а также необходимость программирования на уровне микропрограмм, что является недостатком УУ с "гибкой" логикой по сравнению с УУ с "жесткой" логикой;

- кроме того, УУ с "гибкой" логикой имеет меньшее быстродействие.

6.4 Тактовые генераторы

В процессорах все зависит от порядка выполнения операций. Иногда одна операция должна предшествовать другой, иногда две операции должны происходить одновременно. Для контроля временных параметров в цифровые схемы встраиваются тактовые генераторы, позволяющие обеспечить синхронизацию. Тактовый генератор — это схема, которая вызывает серию импульсов. Все импульсы одинаковы по длительности. Интервалы между последовательными импульсами также одинаковы. Временной интервал между началом одного импульса и началом следующего называется временем такта. Частота импульсов обычно составляет от 100 МГц до 4 ГГц, что соответствует времени такта от 10 до 250 пс. Частота тактового генератора обычно контролируется высокоточным кварцевым генератором.

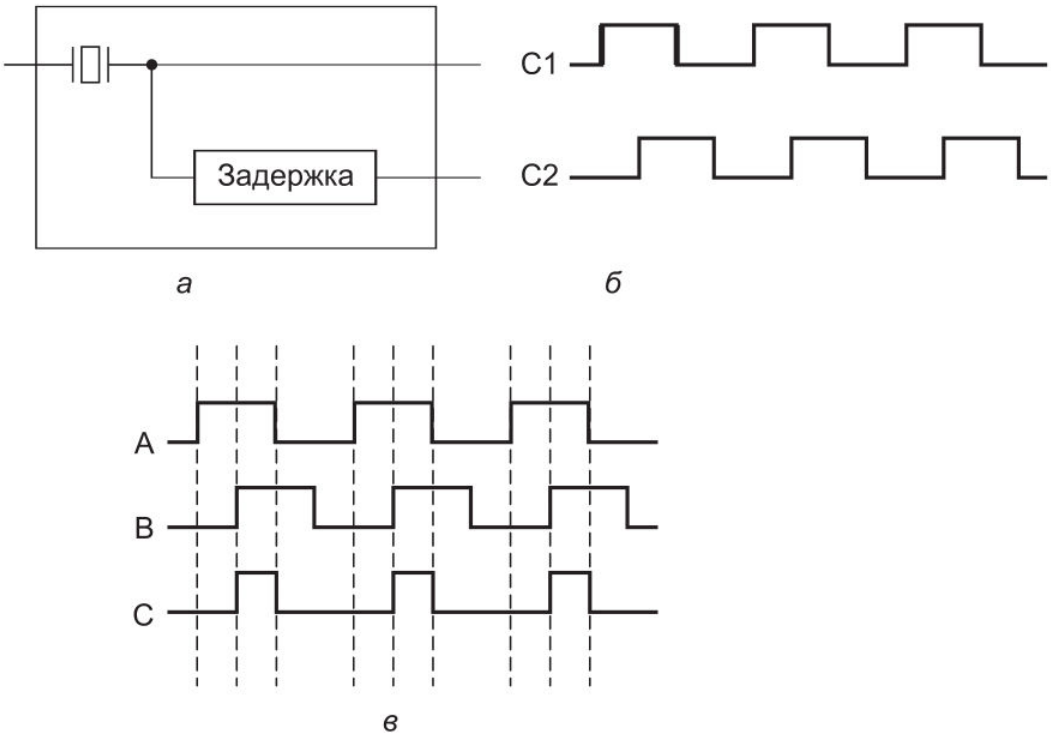

Рис. 6.4.1 Тактовый генератор (а); временная диаграмма тактового генератора (б); порождение асинхронных тактовых импульсов (в)

В компьютере за один такт может произойти множество событий. Если они должны осуществляться в определенном порядке, то такт следует разделить на подтакты. Чтобы достичь лучшего разрешения, чем у основного тактового генератора, нужно сделать ответвление от задающей линии тактового генератора и вставить схему с определенным временем задержки.

Так порождается вторичный сигнал тактового генератора, сдвинутый по фазе относительно первичного (рис. 6.4.1, а). Временная диаграмма, показанная на рис. 6.4.1, б, предлагает четыре точки начала отсчета времени для дискретных событий:

1. Фронт C_1
2. Спад C_1
3. Фронт C_2
4. Спад C_2

Связав различные события с разными перепадами (фронтами и спадами), можно достичь требуемой последовательности выполнения действий. Если в пределах одного такта нужно более четырех точек начала отсчета, можно сделать еще несколько ответвлений от задающей линии с различным временем задержки.

В некоторых схемах важны временные интервалы, а не дискретные моменты времени. Например, некоторое событие может происходить не на фронте импульса, а в любое время, когда уровень импульса C_1 высокий. Другое событие может происходить только в том случае, когда уровень импульса C_2 высокий.

Если необходимо более двух интервалов, нужно предоставить больше линий передачи синхронизирующих импульсов или сделать так, чтобы состояния с высоким уровнем импульса у двух тактовых генераторов частично пересекались во времени. В последнем случае можно выделить 4 отдельных интервала: $!C_1$ И $!C_2$, $!C_1$ И C_2, C_1 И $!C_2$ и C_1 И C_2.

Тактовые генераторы могут быть синхронными. В этом случае время существования импульса с высоким уровнем равно времени существования импульса с низким уровнем (см. рис. 6.4.1, б). Чтобы получить асинхронную серию импульсов (см. сигнал С на рис. 6.4.1, в), нужно сдвинуть сигнал задающего генератора, используя цепь задержки. Затем полученный сигнал соединяется с изначальным сигналом с помощью логической функции И.