Микросхемы процессоров и шины

Вооружившись информацией о микросхемах, тактовых генераторах и микросхемах памяти, мы можем сложить все составные части вместе и начать изучение целых систем. В этом разделе сначала мы рассмотрим процессоры на цифровом логическом уровне, включая **цоколевку** (то есть значение сигналов на различных выводах). Поскольку центральные процессоры тесно связаны с шинами, которые они используют, мы также кратко изложим основные принципы разработки шин. В следующих разделах приводятся подробные примеры центральных процессоров, их шин и взаимодействий между ними.

Микросхемы процессоров

Все современные процессоры помещаются на одной микросхеме, благодаря чему их взаимодействия с остальными частями системы становятся четко определенными. Каждая микросхема процессора содержит набор выводов, через которые происходит обмен информацией с внешним миром. Одни выводы передают сигналы от центрального процессора, другие принимают сигналы от других компонентов, третьи делают то и другое. Изучив функции всех выводов, мы сможем узнать, как процессор взаимодействует с памятью и устройствами ввода-вывода на цифровом логическом уровне.

Выводы микросхемы центрального процессора можно подразделить на три типа: адресные, информационные и управляющие. Эти выводы связаны с соответствующими выводами на микросхемах памяти и микросхемах устройств ввода-вывода через набор параллельных проводов (так называемую шину). Чтобы вызвать команду, центральный процессор сначала посылает в память адрес этой команды по адресным выводам. Затем он задействует одну или несколько линий управления, чтобы сообщить памяти, что ему нужно (например, прочитать слово). Память выдает ответ, помещая требуемое слово на информационные выводы процессора и посылая сигнал о том, что это сделано. Когда центральный процессор получает этот сигнал, он считывает слово и выполняет вызванную команду.

Команда может требовать чтения или записи слов, содержащих данные. В этом случае весь процесс повторяется для каждого дополнительного слова. Как происходит процесс чтения и записи, мы подробно рассмотрим далее. А пока важно понять, что центральный процессор обменивается информацией с памятью и устройствами ввода-вывода, подавая сигналы на выводы и принимая сигналы на входы. Другого способа обмена информацией не существует.

Число адресных выводов и число информационных выводов — два ключевых параметра, которые определяют производительность процессора. Микросхема, содержащая m адресных выводов, может обращаться к 2^m ячейкам памяти. Обычно m равно 16, 32 или 64. Микросхема, содержащая n информационных выводов, может считывать или записывать n-разрядное слово за одну операцию. Обычно n равно 8, 32 или 64. Центральному процессору с 8 информационными выводами понадобится 4 операции, чтобы считать 32-разрядное слово, тогда как процессор, имеющий 32 информационных вывода, может сделать ту же работу в рамках одной операции. Следовательно, микросхема с 32 информационными выводами работает гораздо быстрее, но и стоит гораздо дороже.

Помимо адресных и информационных выводов, каждый процессор содержит управляющие выводы. Эти выводы позволяют регулировать и синхронизировать поток данных к процессору и от него, а также выполнять другие функции. Все процессоры содержат выводы для питания (обычно +1,2 В или +1,5 В), заземления и синхронизирующего сигнала (меандра). Остальные выводы разнятся от процессора к процессору. Тем не менее управляющие выводы можно разделить на несколько основных категорий:

- управление шиной;
- прерывания;
- арбитраж шины;
- сигналы сопроцессора;
- состояние;
- разное.

Далее мы кратко опишем каждую из этих категорий, а когда мы будем рассматривать микросхемы Intel Core i7, TI OMAP4430 и Atmel ATmega168, дадим более подробную информацию. Схема типичного центрального процессора, в котором используются эти типы сигналов, изображена на рис. 3.32.

Выводы управления шиной по большей части представляют собой выходы из центрального процессора в шину (и, следовательно, входы в микросхем памяти и микросхем устройств ввода-вывода). Они позволяют сообщить, что процессор хочет считать информацию из памяти или записать информацию в память или сделать что-нибудь еще.

Выводы прерывания — это входы из устройств ввода-вывода в процессор. В большинстве систем процессор может дать сигнал устройству ввода-вывода начать операцию, а затем приступить к какому-нибудь другому действию, пока устройство ввода-вывода выполняет свою работу. Когда устройство ввода-вывода ее завершит, контроллер ввода-вывода посылает сигнал на один из выводов прерывания, чтобы прервать работу процессора и заставить его обслужить устройство ввода-вывода (например, проверить ошибки ввода-вывода). Некоторые процессоры содержат вывод для подтверждения сигнала прерывания.

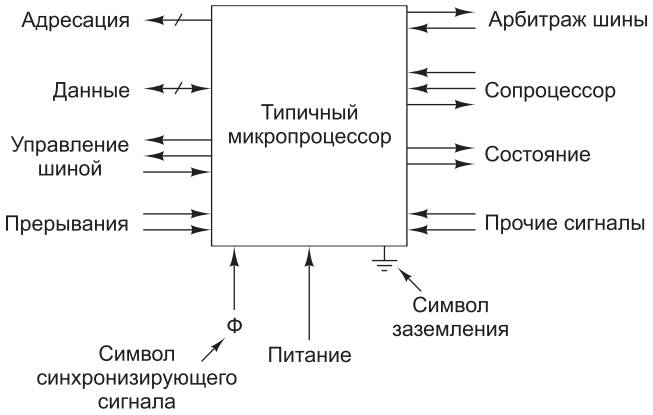


Рис. 3.32. Цоколевка типичного центрального процессора. Стрелками обозначены входные и выходные сигналы, а короткими диагональными линиями — наличие нескольких выводов данного типа. Число этих выводов зависит от модели процессора

Выводы арбитража шины нужны для регулировки потока информации в шине, то есть для исключения таких ситуаций, когда два устройства пытаются воспользоваться шиной одновременно. В плане арбитража центральный процессор считается просто одним из устройств.

Некоторые центральные процессоры могут работать с различными сопроцессорами (например, с графическими процессорами, процессорами для обработки вещественных данных и т. п.). Чтобы обеспечить обмен информацией между процессором и сопроцессором, используются специальные выводы.

Помимо этих выводов, у некоторых процессоров есть дополнительные выводы. Одни из них выдают или принимают информацию о состоянии, другие нужны для перезагрузки компьютера, третьи призваны обеспечивать совместимость со старыми микросхемами устройств ввода-вывода.

Компьютерные шины

Шина — это несколько проводников, соединяющих несколько устройств. Шины можно разделить на категории в соответствии с выполняемыми функциями. Они могут быть внутренними по отношению к процессору и служить для передачи данных в АЛУ и из АЛУ, а могут быть внешними по отношению к процессору и связывать процессор с памятью или устройствами ввода-вывода. Каждый тип шины обладает определенными свойствами и к каждому из них предъявляются определенные требования. В этом и следующих подразделах мы сосредоточимся на шинах, которые связывают центральный процессор с памятью и устройствами ввода-вывода. В следующей главе мы подробно рассмотрим внутренние шины процессора.

Первые персональные компьютеры имели одну внешнюю шину, которая называлась **системной**. Она состояла из нескольких медных проводов (от 50 до 100), которые встраивались в материнскую плату. На материнской плате на одинаковых расстояниях друг от друга находились разъемы для микросхем памяти и устройств ввода-вывода. Современные персональные компьютеры обычно содержат специальную шину между центральным процессором и памятью и по

крайней мере еще одну шину для устройств ввода-вывода. На рис. 3.33 изображена система с одной шиной памяти и одной шиной ввода-вывода.

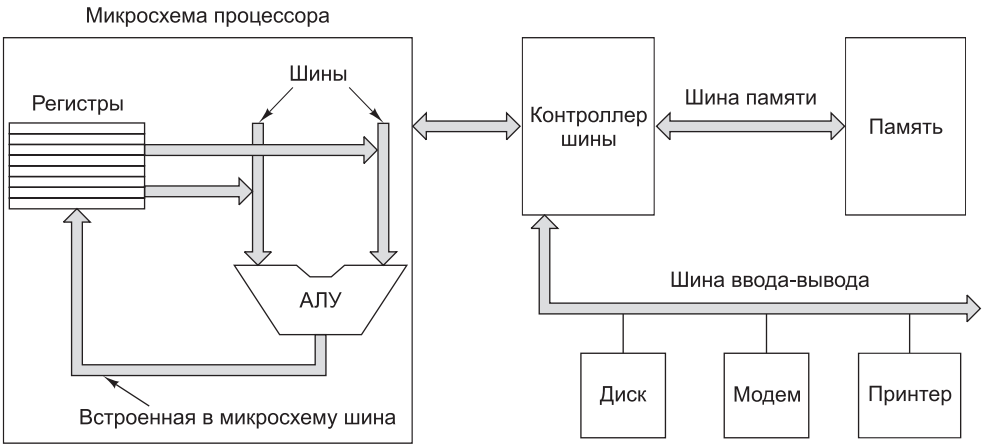


Рис. 3.33. Компьютерная система с несколькими шинами

В литературе шины обычно изображаются в виде жирных стрелок, как показано на этом рисунке. Разница между жирной стрелкой и нежирной стрелкой, через которую проходит короткая диагональная линия с указанием числа битов, небольшая. Когда тип всех битов одинаков, например все адресные или все информационные, рисуется обычная стрелка. Когда включаются адресные линии, линии данных и управления, используется жирная стрелка.

Хотя разработчики процессоров могут использовать любой тип шины для микросхемы, должны быть введены четкие правила о том, как работает шина; и все устройства, связанные с шиной, должны подчиняться этим правилам, чтобы платы, которые выпускаются сторонними производителями, подходили к системной шине. Эти правила называются **протоколом шины**. Кроме того, должны существовать определенные технические требования, чтобы платы от сторонних производителей подходили к направляющим для печатных плат и имели разъемы, соответствующие материнской плате механически, с точки зрения напряжений, синхронизации и т. д. Некоторые шины не имеют механических спецификаций, потому что они спроектированы для использования только с интегральными схемами — например, для соединения компонентов в однокристальных системах (SoC).

Существует целый ряд шин, широко используемых в компьютерном мире, например: Omnibus (PDP-8), Unibus (PDP-11), Multibus (8086), VME (оборудование для физической лаборатории), IBM PC (PC/XT), ISA (PC/AT), EISA (80386), Microchannel (PS/2), Nubus (Macintosh), PCI (различные персональные компьютеры), SCSI (различные персональные компьютеры и рабочие станции), Universal Serial Bus (современные персональные компьютеры), FireWire (бытовая электроника). Может быть, все стало бы намного проще, если бы все шины, кроме одной или двух, исчезли с поверхности земли. К сожалению, стандартизация в этой области кажется очень маловероятной, поскольку во все эти несовместимые системы уже вложено слишком много средств.

Начнем с того, как работают шины. Некоторые устройства, соединенные с шиной, являются активными и могут инициировать передачу информации по шине, тогда как другие являются пассивными и ждут запросов. Активное устройство называется **задающим**, пассивное — **подчиненным**. Когда центральный процессор требует от контроллера диска считать или записать блок информации, центральный процессор действует как задающее устройство, а контроллер диска — как подчиненное. Контроллер диска может действовать как задающее устройство, когда он командует памяти принять слова, которые считал с диска. Несколько типичных комбинаций задающего и подчиненного устройств перечислены в табл. 3.3. Память ни при каких обстоятельствах не может быть задающим устройством.

Таблица 3.3. Примеры задающих и подчиненных устройств

Задающее устройство	**Подчиненное устройство**	**Пример**
Центральный процессор	Память	Вызов команд и данных
Центральный процессор	Устройство ввода-вывода	Инициализация передачи данных
Центральный процессор	Сопроцессор	Передача команды от процессора к сопроцессору
Устройство ввода-вывода	Память	Прямой доступ к памяти
Сопроцессор	Центральный процессор	Вызов сопроцессором операндов из центрального процессора

Двоичным сигналам, которые выдают устройства компьютера, часто не хватает мощности для активизации шины, особенно если она достаточно длинная и если к ней подсоединено много устройств. По этой причине большинство задающих устройств шины обычно связаны с ней через микросхему, которая называется **драйвером шины** и по существу является цифровым усилителем. Сходным образом большинство подчиненных устройств связаны с шиной **приемником шины**. Для устройств, которые могут быть и задающим, и подчиненным устройством, используется **приемопередатчик**, или **трансивер**, **шины**. Эти микросхемы, предназначенные для взаимодействия с шиной, часто являются устройствами с тремя состояниями, что дает им возможность отсоединяться, когда они не нужны. Иногда они подключаются через **открытый коллектор**, что дает сходный эффект. Когда одно или несколько устройств на открытом коллекторе требуют доступа к шине в одно и то же время, результатом является булева операция ИЛИ над всеми этими сигналами. Такое соглашение называется **монтажным ИЛИ**. В большинстве шин одни линии являются устройствами с тремя состояниями, а другие, которым требуется свойство монтажного ИЛИ, — открытым коллектором.

Как и процессор, шина имеет адресные, информационные линии и управляющие линии. Тем не менее между выводами процессора и сигналами шины может и не быть взаимно однозначного соответствия. Например, некоторые процессоры содержат три вывода, которые выдают сигнал чтения из памяти или записи в память, чтения устройства с ввода-вывода, записи на устройство ввода-вывода или выполнения какой-либо другой операции. Обычная шина может содержать одну линию для чтения из памяти, вторую — для записи в память, третью — для

чтения с устройства ввода-вывода, четвертую — для записи на устройство ввода-вывода и т. д. Тогда связывать процессор с такой шиной должна микросхема-декодер, призванная преобразовывать 3-разрядный кодированный сигнал в отдельные сигналы, которые могут управлять линиями шины.

Проектирование и принципы действия шин — это достаточно сложные вопросы, и по этому поводу написан ряд книг [Anderson et al., 2004; Solari and Willse, 2004]. Принципиальными вопросами в разработке являются ширина шины, синхронизация шины, арбитраж шины и функционирование шины. Все эти параметры существенно влияют на пропускную способность шины. В следующих четырех подразделах мы рассмотрим каждый из них.

Ширина шины

Ширина (количество адресных линий) шины — самый очевидный параметр при проектировании. Чем больше адресных линий содержит шина, тем к большему объему памяти может обращаться процессор. Если шина содержит n адресных линий, тогда процессор может использовать ее для обращения к 2^n различным ячейкам памяти. Для памяти большой емкости необходимо много адресных линий. Вроде бы все просто.

Проблема заключается в том, что для широких шин требуется больше проводов, чем для узких. Они занимают больше физического пространства (например, на материнской плате) и для них нужны разъемы большего размера. Все эти факторы делают шину дорогостоящей. Следовательно, необходим компромисс между максимальным объемом доступной памяти и стоимостью системы. Система с шиной, содержащей 64 адресные линии, и памятью в 2^{32} байт будет стоить дороже, чем система с шиной, содержащей 32 адресные линии, и такой же памятью в 2^{32} байт. Дальнейшее расширение не бесплатное.

Многие разработчики систем оказались недальновидными, что привело к неприятным последствиям. Первая модель IBM PC содержала процессор 8088 и 20-разрядную адресную шину (рис. 3.34, *а*). Шина позволяла обращаться к 1 Мбайт памяти.

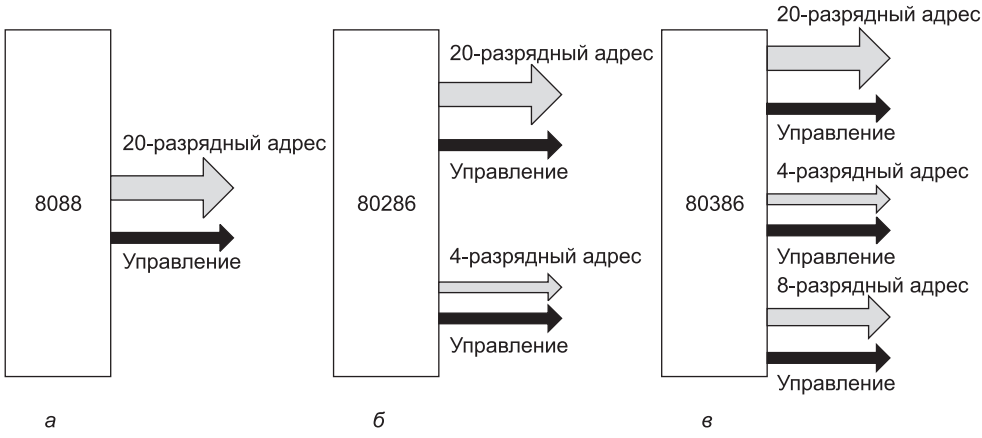


Рис. 3.34. Расширение адресной шины с течением времени

Когда появился следующий процессор (80286), компания Intel решила увеличить адресное пространство до 16 Мбайт, поэтому пришлось добавить еще 4 линии (не нарушая изначальные 20 по причинам совместимости с предыдущими версиями), как показано на рис. 3.34, *б*. К сожалению, пришлось также добавить управляющие линии для новых адресных линий. Когда появился процессор 80386, было добавлено еще 8 адресных линий и, естественно, несколько управляющих линий, как показано на рис. 3.34, *в*. В результате получилась шина EISA. Однако архитектура получилась куда более запутанной, чем если бы с самого начала использовались 32 линии.

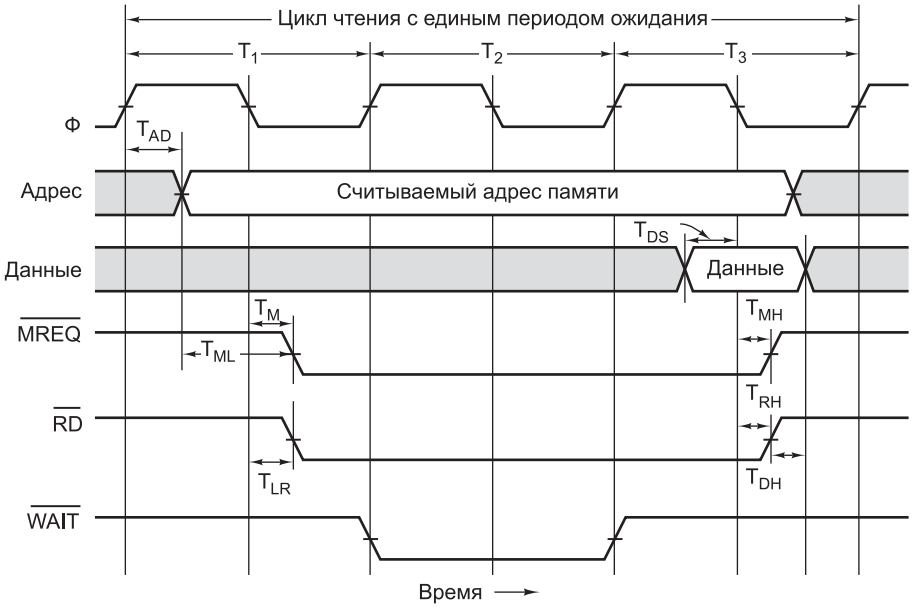
С течением времени увеличивается не только число адресных линий, но и число информационных линий, хотя это происходит по другой причине. Можно увеличить пропускную способность шины двумя способами: сократить время цикла шины (сделать большее количество передач в секунду) или увеличить ширину шины данных (то есть увеличить количество битов, передаваемых за цикл). Можно повысить скорость работы шины, но сделать это довольно сложно, поскольку сигналы на разных линиях передаются с разной скоростью (это явление называется **расфазировкой шины**). Чем быстрее работает шина, тем больше расфазировка.

При увеличении скорости работы шины возникает еще одна проблема: в этом случае она становится несовместимой с предыдущими версиями. Старые платы, разработанные для более медленной шины, не могут работать с новой. Такая ситуация невыгодна для владельцев и производителей старых плат. Поэтому обычно для увеличения производительности просто добавляются новые линии, как показано на рис. 3.34. Как вы понимаете, в этом тоже есть свои недостатки. Компьютер IBM PC и его преемники, например, начали с 8 информационных линий, затем перешли к 16, потом — к 32 линиям, и все это в одной и той же шине.

Чтобы обойти эту проблему, разработчики иногда отдают предпочтение **мультиплексной шине**. В этой шине нет разделения на адресные и информационные линии. В ней может быть, например, 32 линии и для адресов, и для данных. Сначала эти линии используются для адресов, затем — для данных. Чтобы записать информацию в память, нужно сначала передавать в память адрес, а потом — данные. В случае с отдельными линиями адреса и данные могут передаваться вместе. Объединение линий сокращает ширину и стоимость шины, но система работает при этом медленнее. Поэтому проектировщикам приходится взвешивать все за и против, прежде чем сделать выбор.

Синхронизация шины

Шины можно разделить на две категории в зависимости от их синхронизации. **Синхронная шина** содержит линию, которая запускается кварцевым генератором. Сигнал на этой линии представляет собой меандр с частотой обычно от 5 до 133 МГц. Любое действие шины занимает целое число так называемых **циклов шины**. **Асинхронная шина** не содержит задающего генератора. Циклы шины могут быть произвольными и не обязательно одинаковыми для всех пар устройств. Далее мы рассмотрим каждый тип шины отдельно.



а

Символ	Значение	Минимум	Максимум	Единицы измерения
T_{AD}	Задержка выдачи адреса		4	нс
T_{ML}	Промежуток между стабилизацией адреса и установкой сигнала $\overline{MREQ}$	2		нс
T_{M}	Промежуток между спадом синхронизирующего сигнала в цикле T_1 и установкой сигнала $\overline{MREQ}$		3	нс
T_{RL}	Промежуток между спадом синхронизирующего сигнала в цикле T_1 и установкой сигнала $\overline{RD}$		3	нс
T_{DS}	Период передачи данных до спада синхронизирующего сигнала	2		нс
T_{MH}	Промежуток между спадом синхронизирующего сигнала в цикле T_3 и сбросом сигнала $\overline{MREQ}$		3	нс
T_{RH}	Промежуток между спадом синхронизирующего сигнала в цикле T_3 и сбросом сигнала $\overline{RD}$		3	нс
T_{DH}	Период продолжения передачи данных с момента сброса сигнала $\overline{RD}$	0		нс

б

Рис. 3.35. Временна́я диаграмма процесса считывания на синхронной шине (*а*); некоторые временны́е характеристики процесса считывания на синхронной шине (*б*)

Синхронные шины

В качестве примера того, как работает асинхронная шина, рассмотрим временну́ю диаграмму на рис. 3.35. В этом примере мы будем использовать задающий генератор на 100 МГц, который дает цикл шины в 10 нс. Хотя может показаться, что шина работает медленно по сравнению с процессорами на 3 ГГц и выше, не многие современные шины работают быстрее. Например, популярная шина PCI работает с частотой 33 МГц или 66 МГц, а улучшенная (но ныне не используемая) шина PCI-X работала с частотой 133 МГц. О причинах такой низкой скорости современных шин уже рассказывалось: к ним можно отнести такие технические проблемы, как расфазировка шины и необходимость обеспечения совместимости.

В своем примере мы предполагаем, что считывание информации из памяти занимает 15 нс с момента установки адреса. Как мы скоро увидим, для чтения слова понадобится три цикла шины. Первый цикл начинается на фронте отрезка T_1, а третий заканчивается на фронте отрезка T_4, как показано на рис. 3.35. Отметим, что ни один из фронтов и спадов не нарисован вертикальным, потому что ни один электрический сигнал не может изменять свое значение за нулевое время. В нашем примере мы предполагаем, что для изменения сигнала требуется 1 нс. Генератор и линии адреса и данных, а также линии $\overline{\text{MREQ}}$, $\overline{\text{RD}}$, $\overline{\text{WAIT}}$ показаны в том же масштабе времени.

Начало T_1 определяется фронтом генератора. За время T_1 центральный процессор помещает адрес нужного слова на адресные линии. Поскольку адрес представляет собой не одно значение (в отличие от генератора), мы не можем показать его в виде одной линии на схеме. Вместо этого мы показали его в виде двух линий с пересечениями там, где этот адрес меняется. Серый цвет на схеме показывает, что в этот момент не важно, какое значение принял сигнала. Используя то же соглашение, мы видим, что содержание линий данных не имеет значения до отрезка T_3.

После того как у адресных линий появляется возможность приобрести новое значение, устанавливаются сигналы $\overline{\text{MREQ}}$ и $\overline{\text{RD}}$. Первый указывает, что осуществляется доступ к памяти, а не к устройству ввода-вывода, а второй — что осуществляется чтение, а не запись. Поскольку после установки адреса считывание информации из памяти занимает 15 нс (часть первого цикла), память не может передать требуемые данные за период T_2. Чтобы центральный процессор не ожидал поступления данных, память устанавливает сигнал $\overline{\text{WAIT}}$ в начале отрезка T_2. Это означает ввод **периодов ожидания** (дополнительных циклов шины) до тех пор, пока память не сбросит сигнал $\overline{\text{WAIT}}$. В нашем примере вводится один период ожидания (T_2), поскольку память работает слишком медленно. В начале отрезка T_3, когда есть уверенность в том, что память получит данные в течение текущего цикла, сигнал $\overline{\text{WAIT}}$ сбрасывается.

Во время первой половины отрезка T_3 память помещает данные на информационные линии. На спаде отрезка T_3 центральный процессор стробирует (то есть считывает) информационные линии, сохраняя их значения во внутреннем регистре. Считав данные, центральный процессор сбрасывает сигналы $\overline{\text{MREQ}}$ и $\overline{\text{RD}}$. В случае необходимости на следующем фронте может начаться еще один цикл памяти. Эта последовательность может повторяться бесконечно.

В хронометражной спецификации на рис. 3.35, *б* используются 8 условных обозначений. T_{AD}, например, — это временной интервал между фронтом T_1 и установкой адресных линий. В соответствии с требованиями синхронизации $T_{AD} \leq 4$ нс. Это значит, что производитель процессора гарантирует, что во время любого цикла считывания центральный процессор сможет выдать требуемый адрес в пределах 4 нс от середины фронта T_1.

Условия синхронизации также требуют, чтобы данные поступали на информационные линии по крайней мере за 2 нс (T_{DS}) до спада T_3, чтобы дать данным время установиться до того, как процессор начнет их стробировать. Сочетание ограничений на T_{AD} и T_{DS} означает, что в худшем случае в распоряжении памяти будет только 25 — 4 — 2 = 19 нс с момента появления адреса и до момента, когда нужно выдавать данные. Поскольку достаточно 10 нс, память даже в самом худшем случае может всегда ответить за период T_3. Если памяти для считывания требуется 20 нс, то необходимо ввести второй период ожидания, и тогда память ответит в течение T_4.

Требования синхронизации гарантируют, что адрес будет установлен по крайней мере за 2 нс до того, как появится сигнал $\overline{\text{MREQ}}$. Это время может быть важно в том случае, если $\overline{\text{MREQ}}$ инициирует выбор элемента памяти, поскольку некоторые типы памяти требуют определенного времени на установку адреса до выбора элемента памяти. Ясно, что разработчику системы не следует выбирать микросхему памяти, которой нужно 3 нс на подготовку.

Ограничения на T_M и T_{RL} означают, что сигналы $\overline{\text{MREQ}}$ и $\overline{\text{RD}}$ будут установлены в пределах 3 нс от спада T_1. В худшем случае у микросхемы памяти после установки сигналов $\overline{\text{MREQ}}$ и $\overline{\text{RD}}$ останется всего 10 + 10 — 3 — 2 = 15 нс на передачу данных по шине. Это ограничение вводится дополнительно по отношению к интервалу в 15 нс и не зависит от него.

Интервалы T_{MH} и T_{RH} определяют, сколько времени требуется на отмену сигналов $\overline{\text{MREQ}}$ и $\overline{\text{RD}}$ после того, как данные стробированы. Наконец, интервал T_{DH} определяет, сколько времени память должна держать данные на шине после снятия сигнала $\overline{\text{RD}}$. В нашем примере при данном процессоре память может удалить данные с шины, как только сбрасывается сигнал $\overline{\text{RD}}$; в случае других процессоров данные могут сохраняться еще некоторое время.

Необходимо подчеркнуть, что наш пример представляет собой весьма упрощенную версию реальных временны́х ограничений. В действительности таких ограничений гораздо больше. Тем не менее этот пример наглядно демонстрирует, как работает синхронная шина.

Отметим, что сигналы управления могут задаваться низким или высоким напряжением. Что является более удобным в каждом конкретном случае, должен решать разработчик, хотя, по существу, выбор произволен. Такую свободу выбора можно назвать «аппаратным» аналогом ситуации, при которой программист может представить свободные дисковые блоки в битовом отображении как в виде нулей, так и в виде единиц.

Асинхронные шины

Хотя использовать синхронные шины благодаря дискретным временны́м интервалам достаточно удобно, здесь все же есть некоторые проблемы. Например, если процессор и память способны закончить передачу за 3,1 цикла, они вынуждены продлить ее до 4,0 цикла, поскольку неполные циклы запрещены.

Еще хуже то, что если однажды был выбран определенный цикл шины и в соответствии с ним разработана память и карты ввода-вывода, то в будущем трудно делать технологические усовершенствования. Например, предположим, что через несколько лет после выпуска системы, изображенной на рис. 3.35, появилась новая память с временем доступа не в 15, а в 8 нс. Это время позволяет избавиться от периода ожидания и увеличить скорость работы машины. А теперь представим, что появилась память с временем доступа в 4 нс. При этом улучшения производительности уже не будет, поскольку в данной разработке минимальное время чтения — 2 цикла.

Если синхронная шина соединяет ряд устройств, одни из которых работают быстро, а другие медленно, шина подстраивается под самое медленное устройство, а более быстрые не могут использовать свой потенциал полностью.

По этой причине были разработаны асинхронные шины, то есть шины без задающего генератора (рис. 3.36). Работа асинхронной шины не привязывается к генератору. Когда задающее устройство устанавливает адрес, сигнал $\overline{\text{MREQ}}$, $\overline{\text{RD}}$ или любой другой требуемый сигнал, он выдает специальный синхронизирующий сигнал $\overline{\text{MSYN}}$ (Master SYNchronization). Когда подчиненное устройство получает этот сигнал, оно начинает выполнять свою работу настолько быстро, насколько это возможно. Когда работа заканчивается, подчиненное устройство выдает сигнал $\overline{\text{SSYN}}$ (Slave SYNchronization).

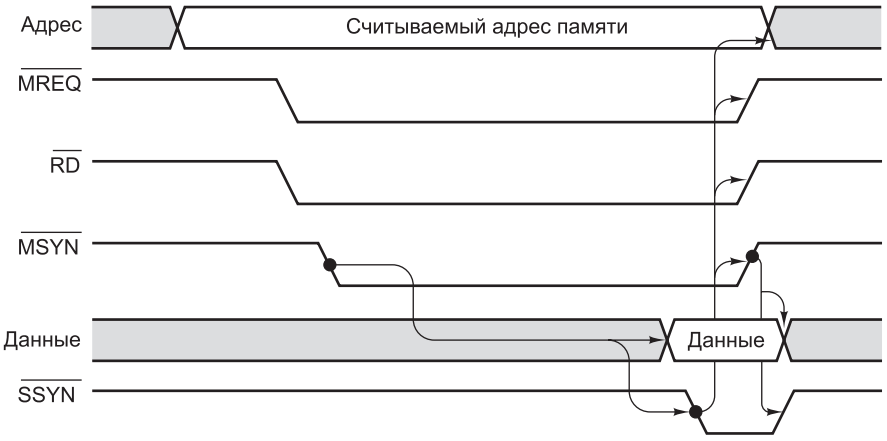


Рис. 3.36. Работа асинхронной шины

Сигнал $\overline{\text{SSYN}}$ сообщает задающему устройству, что данные доступны. Он фиксирует их, а затем сбрасывает адресные линии вместе с сигналами $\overline{\text{MREQ}}$, $\overline{\text{RD}}$ и $\overline{\text{MSYN}}$. Сброс сигнала $\overline{\text{MSYN}}$ означает для подчиненного устройства, что цикл закончен, поэтому устройство сбрасывает сигнал $\overline{\text{SSYN}}$, и все возвращается к первоначальному состоянию, когда все сигналы сброшены.

Стрелочки на временны́х диаграммах асинхронных шин (а иногда и синхронных шин) показывают причину и следствие какого-либо действия (см. рис. 3.36). Установка сигнала $\overline{\text{MSYN}}$ приводит к включению информационных линий, а также к установке сигнала $\overline{\text{SSYN}}$. Установка сигнала $\overline{\text{SSYN}}$, в свою очередь, вызывает отключение адресных линий, а также линий $\overline{\text{MREQ}}$, $\overline{\text{RD}}$ и $\overline{\text{MSYN}}$.

Наконец, сброс сигнала $\overline{\text{MSYN}}$ вызывает сброс сигнала $\overline{\text{SSYN}}$, и на этом процесс считывания заканчивается.

Набор таких взаимообусловленных сигналов называется **полным квитированием**. Здесь, в сущности, наблюдается 4 события:

1. Установка сигнала $\overline{\text{MSYN}}$.
2. Установка сигнала $\overline{\text{SSYN}}$ в ответ на сигнал $\overline{\text{MSYN}}$.
3. Сброс сигнала $\overline{\text{MSYN}}$ в ответ на сигнал $\overline{\text{SSYN}}$.
4. Сброс сигнала $\overline{\text{SSYN}}$ в ответ на сброс сигнала $\overline{\text{MSYN}}$.

Разумеется, взаимообусловленность сигналов не является синхронной. Каждое событие вызывается предыдущим событием, а не импульсами генератора. Если какая-то пара устройств (задающее и подчиненное) работает медленно, это никак не влияет на другую пару устройств, которая может работать гораздо быстрее.

Преимущества асинхронной шины очевидны, хотя на самом деле большинство шин являются синхронными. Дело в том, что синхронную систему построить проще, чем асинхронную. Центральный процессор просто выдает сигналы, а память просто реагирует на них. Здесь нет никакой причинно-следственной связи, а если компоненты выбраны удачно, все работает и без квитирования. Кроме того, в разработку синхронных шин вложено очень много ресурсов.

Арбитраж шины

До этого момента мы неявно предполагали, что существует только одно задающее устройство шины — центральный процессор. В действительности микросхемы ввода-вывода могут становиться задающими устройствами при считывании информации из памяти и записи информации в память. Кроме того, они могут вызывать прерывания. Сопроцессоры также могут становиться задающими устройствами шины. Возникает вопрос: «Что происходит, когда задающим устройством шины становятся два или более устройств одновременно?» Чтобы предотвратить хаос, который может при этом возникнуть, нужен специальный механизм — так называемый **арбитраж шины**.

Арбитраж может быть централизованным или децентрализованным. Рассмотрим сначала централизованный арбитраж. Простой пример централизованного арбитража показан на рис. 3.37, *а*. В данном примере один арбитр шины определяет, чья очередь следующая. Часто механизм арбитража встраивается в микросхему процессора, но иногда используется отдельная микросхема. Шина содержит одну линию запроса (монтажное ИЛИ), которая может запускаться одним или несколькими устройствами в любое время. Арбитр не может определить, сколько устройств запрашивают шину. Он может определить только факт наличия или отсутствия запросов.

Когда арбитр обнаруживает запрос шины, он устанавливает линию предоставления шины. Эта линия последовательно связывает все устройства ввода-вывода (как в елочной гирлянде). Когда физически ближайшее к арбитру устройство получает сигнал предоставления шины, это устройство проверяет, нет ли запроса шины. Если запрос есть, устройство пользуется шиной, но не распространяет сигнал предоставления дальше по линии. Если запроса нет, устройство передает сигнал предоставления шины следующему устройству. Это устройство тоже про-

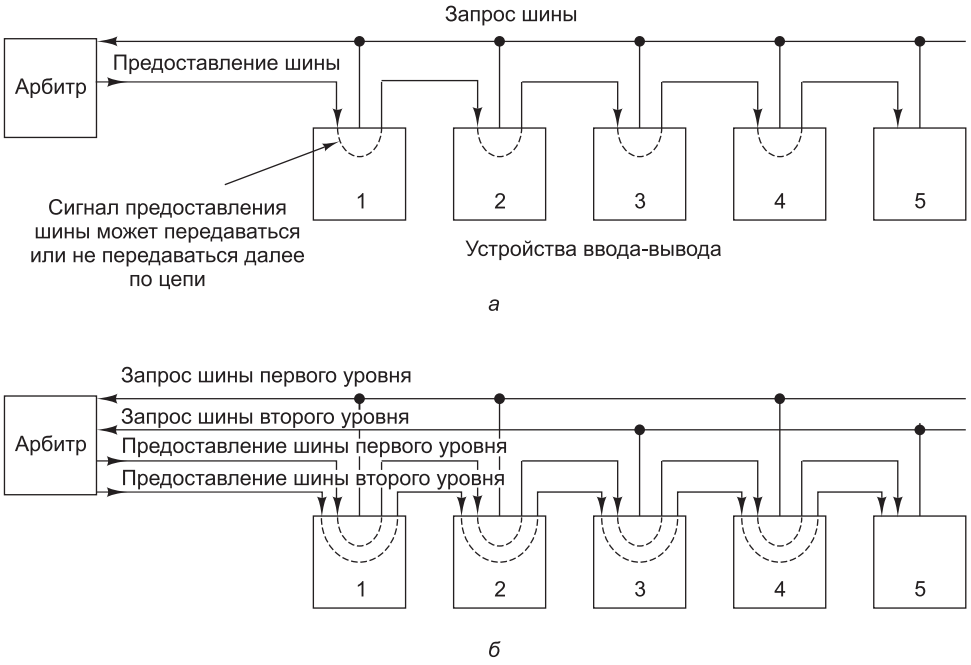


Рис. 3.37. Одноуровневый централизованный арбитраж шины с последовательным опросом (*а*); двухуровневый централизованный арбитраж (*б*)

веряет, есть ли запрос, и действует соответствующим образом в зависимости от наличия или отсутствия запроса. Передача сигнала предоставления шины продолжается до тех пор, пока какое-нибудь устройство не воспользуется предоставленной шиной. Такая система называется **системой последовательного опроса**. При этом приоритеты устройств зависят от того, насколько близко они находятся к арбитру. Ближайшее к арбитру устройство обладает наивысшим приоритетом.

Чтобы приоритеты устройств не зависели от расстояния от арбитра, в некоторых шинах поддерживается несколько уровней приоритета. На каждом уровне приоритета есть линия запроса шины и линия предоставления шины. На рис. 3.37, *б* изображено 2 уровня (хотя в действительности шины обычно поддерживают 4, 8 или 16 уровней). Каждое устройство связано с одним из уровней запроса шины, причем чем выше уровень приоритета, тем больше устройств привязано к этому уровню. На рис. 3.37, *б* можно видеть, что устройства 1, 2 и 4 обладают приоритетом уровня 1, а устройства 3 и 5 — приоритетом уровня 2.

Если одновременно запрашивается несколько уровней приоритета, арбитр предоставляет шину самому высокому уровню. Среди устройств одинакового приоритета реализуется система последовательного опроса. На рис. 3.37, *б* видно, что в случае конфликта устройство 2 «побеждает» устройство 4, а устройство 4 «побеждает» устройство 3. Устройство 5 имеет низший приоритет, поскольку оно находится в самом конце самого нижнего уровня.

Следует заметить, что с технической точки зрения линия предоставления шины уровня 2 не обязательно должна последовательно связывать устройства 1 и 2, поскольку они не могут посылать на нее запросы. Однако гораздо проще

провести все линии предоставления шины через все устройства, чем соединять устройства особым образом в зависимости от их приоритетов.

Некоторые арбитры содержат третью линию, которая устанавливается, как только устройство принимает сигнал предоставления шины, и получает шину в свое распоряжение. Как только эта линия подтверждения приема устанавливается, линии запроса и предоставления шины могут быть сброшены. В результате другие устройства могут запрашивать шину, пока первое устройство ее использует. К тому моменту, когда закончится текущая передача, следующее задающее устройство уже будет выбрано. Это устройство может начать работу, как только будет сброшена линия подтверждения приема. С этого момента начинается следующий цикл арбитража. Такая структура требует дополнительной линии и большего количества логических схем в каждом устройстве, но зато при этом циклы шины используются рациональнее.

В системах, где память связана с главной шиной, центральный процессор должен конкурировать со всеми устройствами ввода-вывода практически на каждом цикле шины. Чтобы решить эту проблему, можно предоставить центральному процессору самый низкий приоритет. При этом шина будет предоставляться процессору только в том случае, если она не нужна ни одному другому устройству. Центральный процессор всегда может подождать, а устройства ввода-вывода должны получить доступ к шине как можно быстрее, чтобы не потерять данные. Например, диски, вращающиеся с высокой скоростью, не могут ждать. Во многих современных компьютерах для решения этой проблемы память помещается на одну шину, а устройства ввода-вывода — на другую, поэтому им не приходится завершать работу, чтобы предоставить доступ к шине.

Возможен также децентрализованный арбитраж шины. Например, компьютер может содержать 16 приоритетных линий запроса шины. Когда устройству нужна шина, оно устанавливает свою линию запроса. Все устройства отслеживают все линии запроса, поэтому в конце каждого цикла шины каждое устройство может определить, обладает ли оно в данный момент наивысшим приоритетом и, следовательно, разрешено ли ей пользоваться шиной в следующем цикле. Такой метод требует большего количества линий, но зато избавляет от потенциальных затрат ресурсов на использование арбитра. В этом случае число устройств ограничивается числом линий запроса.

При другом типе децентрализованного арбитража используются только три линии независимо от того, сколько устройств имеется в наличии (рис. 3.38). Первая линия — монтажное ИЛИ. Она требуется для запроса шины. Вторая линия называется BUSY и означает занятость. Она запускается текущим задающим устройством шины. Третья линия служит для арбитража шины. Она

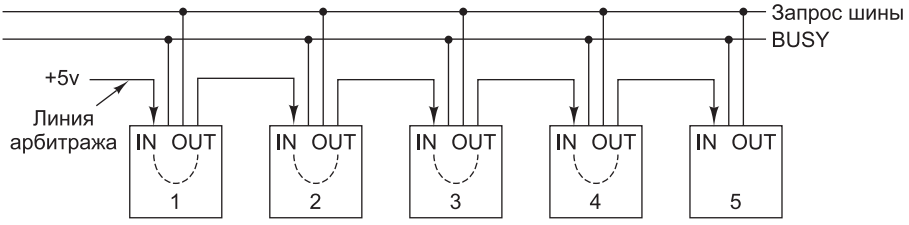


Рис. 3.38. Децентрализованный арбитраж шины

последовательно соединяет все устройства. Начало цепи связано с источником питания с напряжением 5 В.

Когда шина не требуется ни одному из устройств, линия арбитража передает сигнал всем устройствам. Чтобы получить доступ к шине, устройство сначала проверяет, свободна ли шина и установлен ли сигнал арбитража IN. Если сигнал IN не установлен, устройство не может стать задающим устройством шины. В этом случае оно сбрасывает сигнал OUT. Если сигнал IN установлен, устройство также сбрасывает сигнал OUT, в результате чего следующее устройство не получает сигнала IN и, в свою очередь, сбрасывает сигнал OUT. Следовательно, все следующие по цепи устройства не получают сигнал IN и сбрасывают сигнал OUT. В результате остается только одно устройство, у которого сигнал IN установлен, а сигнал OUT сброшен. Оно становится задающим устройством шины, устанавливает линию BUSY и сигнал OUT, после чего начинает передачу данных.

Немного поразмыслив, можно обнаружить, что из всех устройств, которым нужна шина, доступ к шине получает самое левое. Такая система напоминает систему последовательного опроса, только в данном случае нет арбитра, поэтому она стоит дешевле и работает быстрее. К тому же не возникает проблем со сбоями арбитра.

Принципы работы шины

До этого момента мы обсуждали только обычные циклы шины, когда задающее устройство (обычно центральный процессор) считывает информацию из подчиненного устройства (обычно из памяти) или записывает в него информацию. Однако существуют еще несколько типов циклов шины. Давайте рассмотрим некоторые из них.

Обычно за раз передается одно слово. При использовании кэш-памяти желательно сразу вызывать всю строку кэш-памяти (то есть 16 последовательных 64-разрядных слов). Однако часто передача блоками может быть более эффективна, чем такая последовательная передача информации. Когда начинается чтение блока, задающее устройство сообщает подчиненному устройству, сколько слов нужно передать (например, помещая общее число слов на информационные линии в период T_1). Вместо того чтобы выдать в ответ одно слово, задающее устройство выдает одно слово в течение каждого цикла до тех пор, пока не будет передано требуемое количество слов. На рис. 3.39 изображена такая же схема, как и на рис. 3.35, только с дополнительным сигналом $\overline{\text{BLOCK}}$, который указывает, что запрашивается передача блока. В данном примере считывание блока из четырех слов занимает 6 циклов вместо 12-ти.

Существуют также другие типы циклов шины. Например, если речь идет о системах с двумя или несколькими центральными процессорами на одной шине, нужно быть уверенным, что в конкретный момент только один центральный процессор может использовать определенную структуру данных в памяти. Чтобы упорядочить этот процесс, в памяти должна содержаться переменная, которая принимает значение 0, когда центральный процессор использует структуру данных, и 1, когда структура данных не используется. Если центральному процессору нужно получить доступ к структуре данных, он должен считать переменную, и если она равна 0, придать ей значение 1. Проблема заключается в том, что два

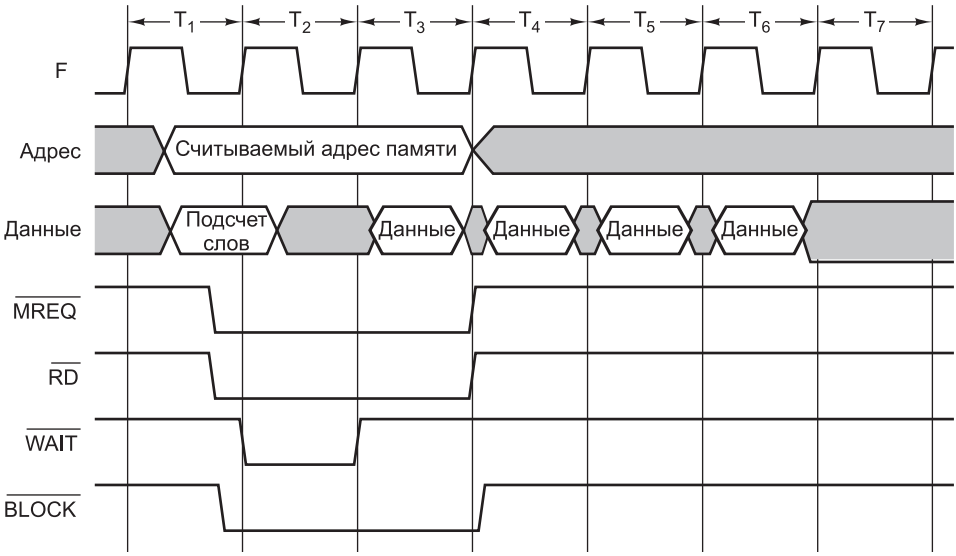


Рис. 3.39. Передача блока данных

центральных процессора могут считывать переменную на последовательных циклах шины. Если каждый процессор обнаружит, что переменная равна 0, а затем поменяет значение переменной на 1, как будто только он один использует эту структуру данных, то такая последовательность событий приведет к хаосу.

Чтобы не допустить подобную ситуацию, в мультипроцессорных системах предусмотрен специальный цикл шины, который дает возможность любому процессору считать слово из памяти, проверить и изменить его, а затем записать обратно в память; весь этот процесс происходит без освобождения шины. Такой цикл не дает возможности другим центральным процессорам использовать шину и, следовательно, мешать работе первого процессора.

Еще один важный цикл шины — цикл обработки прерываний. Когда центральный процессор командует устройству ввода-вывода произвести какое-то действие, он ожидает прерывания после завершения работы. Для сигнала прерывания нужна шина.

Поскольку может сложиться ситуация, когда несколько устройств одновременно захотят выполнить прерывание, здесь имеют место те же проблемы разрешения конфликтных ситуаций, что и в обычных циклах шины. Чтобы избежать таких проблем, нужно каждому устройству приписать определенный приоритет и для распределения приоритетов поддерживать централизованный арбитраж. Для этих целей существует стандартный, широко используемый интерфейс прерываний. В компьютерах IBM PC и последующих моделях для этого служит микросхема Intel 8259A. Она изображена на рис. 3.40.

До восьми контроллеров ввода-вывода могут быть непосредственно связаны с восемью входами IR*x* (Interrupt Request — запрос прерывания) микросхемы 8259A. Когда любое из этих устройств решит произвести прерывание, оно запускает свою линию входа. При активизации одного или нескольких входов контроллер 8259A выдает сигнал INT (INTerrupt — прерывание), который подается

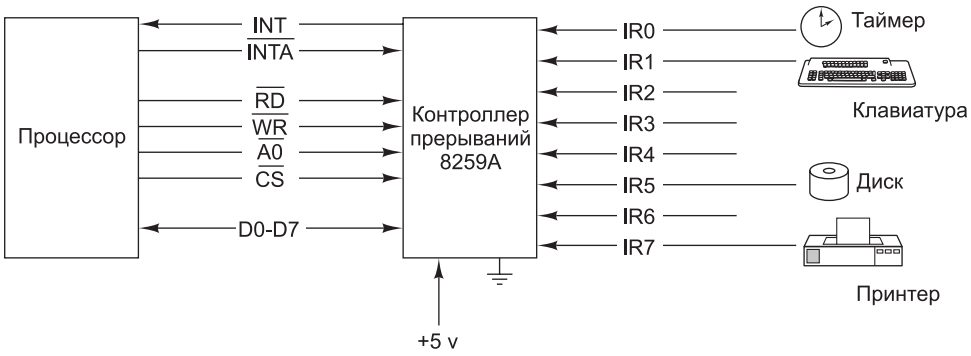


Рис. 3.40. Контроллер прерываний 8259A

на соответствующий вход центрального процессора. Если центральный процессор способен обработать прерывание, он посылает микросхеме 8259A импульс через вывод INTA (INTerrupt Acknowledge — подтверждение прерывания). В этот момент микросхема 8259A должна определить, на какой именно вход поступил сигнал прерывания. Для этого она помещает номер входа на информационную шину. Эта операция требует особого цикла шины. Центральный процессор использует этот номер для обращения к таблице указателей, которую называют таблицей **векторов прерываний**, чтобы найти адрес процедуры обработки этого прерывания.

Микросхема 8259A содержит несколько регистров, которые центральный процессор может считывать и записывать, используя обычные циклы шины и выводы $\overline{\text{RD}}$ (ReaD — чтение), $\overline{\text{WR}}$ (WRite — запись), $\overline{\text{CS}}$ (Chip Select — выбор элемента памяти) и $\overline{\text{A0}}$. Когда программное обеспечение обработало прерывание и готово получить следующее, оно записывает специальный код в один из регистров, который вызывает сброс сигнала INT микросхемой 8259A, если не появляется другое прерывание. Регистры также могут записываться для того, чтобы перевести микросхему 8259A в один из нескольких режимов, и для выполнения некоторых других функций.

При наличии более 8 устройств ввода-вывода, микросхемы 8259A могут соединяться каскадом. В самой экстремальной ситуации все 8 входов могут быть связаны с выходами еще 8 микросхем 8259A, соединяя до 64 устройств ввода-вывода в двухступенчатую систему обработки прерываний. Контроллер-концентратор ввода/вывода Intel ICH10 I/O, одна из микросхем чипсета Core i7, содержат два контроллера прерываний 8259A. Таким образом, ICH10 имеет 15 внешних прерываний — на 1 меньше 16 прерываний двух контроллеров 8259A, так как одно из прерываний используется для каскадного подключения второго контроллера 8259A. Микросхема 8259A содержит несколько выводов для каскадного соединения, но мы их опустили ради простоты. В наши дни 8259A является составной частью другой микросхемы.

Хотя приведенное описание никоим образом не исчерпывает всех вопросов разработки шин, оно дает достаточно информации для общего понимания принципов работы шины и принципов взаимодействия с шиной центрального процессора. Теперь мы перейдем от общего к частному и рассмотрим несколько конкретных примеров процессоров и их шин.

Примеры центральных процессоров

В этом разделе мы рассмотрим процессоры Intel Core i7, TI OMAP4430 и Atmel ATmega168 на уровне аппаратного обеспечения.

Intel Core i7

Core i7 — прямой потомок процессора 8088, который использовался в первой модели IBM PC. Презентация Core i7 состоялась в ноябре 2008 года. Публике было представлено четырехпроцессорное ЦПУ с 731 млн транзисторов, частотой до 3,2 ГГц и шириной строки 45 нанометра. Понятие «ширина строки» обозначает ширину проводников между транзисторами (и одновременно определяет размер самих транзисторов). Чем меньше эта величина, тем больше транзисторов умещается на одной микросхеме. По сути, закон Мура прогнозирует способность инженеров к дальнейшему уменьшению ширины строки. Помимо прочего, уменьшение этой величины позволяет повысить тактовую частоту. Для сравнения, диаметр человеческого волоса составляет 20–100 мкм (причем светлые волосы тоньше темных).

Исходный выпуск архитектуры Core i7 базировался на архитектуре «Nahalem», однако новые версии Core i7 строятся на базе более новой архитектуры «Sandy Bridge». Термином «архитектура» в этом контексте обозначается внутренняя организация центрального процессора, которой часто присваивается кодовое название. Обычно проектировщики компьютерных архитектур — люди серьезные, но иногда они придумывают для своих проектов очень остроумные кодовые названия. Например, архитектуры серии AMD K должны были разрушить позиции Intel на рынке процессоров для настольных систем, казавшиеся неуязвимыми. Для процессоров серии K было выбрано кодовое название «Kryptonite» — название единственного вещества, которое могло повредить Супермену, остроумный намекало на доминирование Intel.

Новая версия Core i7 на базе архитектуры «Sandy-Bridge» увеличилась до 1,16 млрд транзисторов. Она работает на скорости 3,5 ГГц с шириной строки 32 нанометра. Хотя Core i7 очень сильно отличается от процессора 8088 с его 29 000 транзисторов, он полностью совместим с 8088 и может выполнять двоичные программы, написанные для 8088 (не говоря уже о программах для всех процессоров, появившихся между Core i7 и 8088).

С точки зрения программного обеспечения Core i7 представляет собой 64-разрядную машину. Он поддерживает ту же стандартную промышленную архитектуру (ISA), что и процессоры 80386, 80486, Pentium, Pentium II, Pentium Pro, Pentium III и Pentium 4, включая те же регистры, те же команды и такую же встроенную систему обработки значений с плавающей точкой стандарта IEEE 754. Помимо этого, в Core i7 имеются новые команды, предназначенные в первую очередь для криптографических операций.

Core i7 является многоядерным процессором; таким образом, кремниевая подложка содержит несколько процессоров. Он продается с разным числом внутренних процессов — от 2 до 6 (причем в ближайшем будущем их число должно увеличиться). Если программисты пишут параллельную программу с использованием потоков и блокировок, организация параллельного выполнения на несколь-

ких процессорах обеспечит существенный выигрыш по скорости. поддерживается технология **гиперпоточности**, позволяющая нескольким аппаратным потокам быть активными одновременно. Гиперпоточность позволяет осуществлять аппаратное переключение потоков во время очень коротких задержек (например, промахов кэша). Программное переключение потоков может происходить только во время очень длинных задержек (например, сбоев страниц), поскольку для его реализации требуются сотни тактов.

На уровне микроархитектуры Core i7 базируется на архитектуре своих предшественников Core 2 и Core 2 Duo. Процессор Core i7 может выполнять до четырех команд одновременно, что позволяет рассматривать его как 4-кратную суперскалярную машину. Микроархитектуру Core i7 мы обсудим в главе 4.

В процессорах Core i7 используется трехуровневый кэш. Каждый процессор Core i7 имеет 32-килобайтный кэш данных первого уровня (L1) и 32-килобайтный кэш команд первого уровня. У каждого ядра также имеется собственный 256-килобайтный кэш второго уровня (L2). Кэш второго уровня унифицирован, то есть позволяет хранить комбинацию команд и данных. Все ядра совместно используют один унифицированный кэш третьего уровня (L3), размер которого составляет от 4 до 15 Мбайт в зависимости от модели процессора. Трехуровневое кэширование значительно улучшает производительность процессора, но за счет возрастания стоимости кремниевых компонентов, так как у процессоров Core i7 общий объем кэша на одной подложке не может превышать 17 Мбайт.

Поскольку все микросхемы Core i7 содержат несколько процессоров с собственными кэшами данных, при изменении одним из процессоров слова, размещенного в его приватном кэше, могут возникать трудности. Если, предположим, другой процессор попытается считать это слово из памяти, он получит устаревшее значение, поскольку между изменением слова и его записью в память проходит некоторое время. В целях поддержания согласованности данных в памяти каждый ЦП в мультипроцессорной системе **следит** за шиной памяти на предмет поиска запросов на кэшированные слова. В случае обнаружения подобного рода запроса процессор предоставляет необходимые данные до того, как память передаст их другим потребителям. Технологию слежения мы рассмотрим в главе 8.

В системах с процессором Core i7 используются две внешние шины, обе они синхронные. Шина памяти DDR3 служит для доступа к главному динамическому ОЗУ; шина PCI Express — для взаимодействия с устройствами ввода-вывода. Высокопроизводительные версии Core i7 содержат несколько шин памяти и PCI Express, а также порт QPI (Quick Path Interconnect). Порт QPI связывает процессор с внешним мультипроцессорным соединением, что открывает возможность построения систем, в которых установлено более шести процессоров. Порт QPI отправляет и получает **запросы когерентности кэша**, а также другие управляющие сообщения для мультипроцессорных систем — например, межпроцессорные прерывания.

Основная проблема Core i7, как и у всех современных процессоров для настольных систем, заключается в объемах потребляемой мощности и выделяемого тепла. Чтобы избежать повреждения кремниевых компонентов, необходимо отводить тепло от процессора сразу же после его образования. Процессоры Core i7 в зависимости от частоты и модели потребляют от 17 до 1502 Вт. Поэтому

Intel пребывает в постоянном поиске новых решений, которые позволили бы урегулировать проблему тепловыделения. Технологии охлаждения и теплопроводящие корпуса играют важную роль для защиты от выгорания кремниевых компонентов.

Микросхемы Core i7 поставляются в квадратном корпусе LGA с длиной стороны 37,5 мм. На нижней плоскости микросхемы находится 1155 площадок, из которых 286 используются для подачи питания, а 360 заземляются в целях шумоподавления. Площадки размещены в виде матрицы 40 × 40, причем ее центральный сегмент 17 × 25 не заполнен. Кроме того, по периметру асимметрично пропущены еще 20 площадок, за счет чего исключается возможность неправильной установки микросхемы в гнезде. Физическую компоновку Core i7 иллюстрирует рис. 3.41.

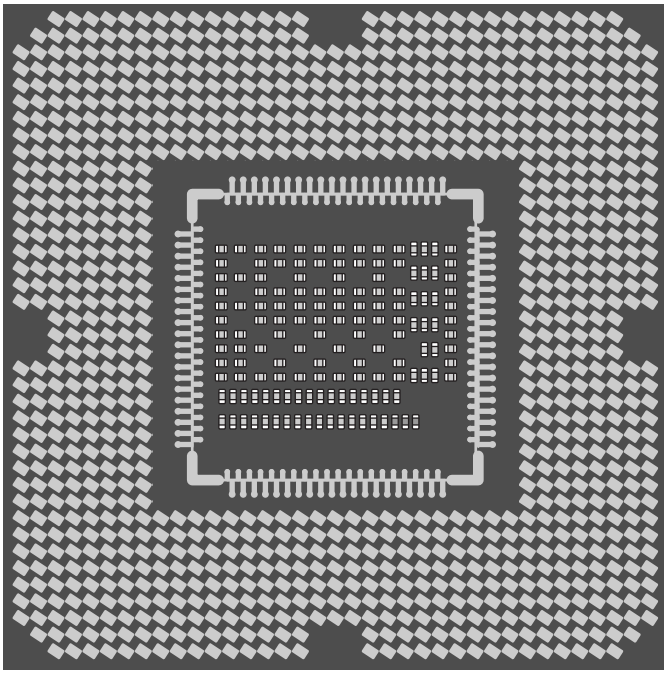


Рис. 3.41. Компоновка Core i7

Микросхема снабжена креплением для радиатора, который рассеивает тепло, и вентилятора, который охлаждает процессор. Чтобы получить некоторое представление о том, что собой представляет величина 150 Вт, поднесите руку к включенной электрической лампочке мощностью 150 Вт (только не дотрагивайтесь до нее). Вот такое количество тепла нужно рассеивать постоянно. Соответственно, когда Core i7 потеряет свои рабочие характеристики как процессор, он вполне сгодится в качестве нагревателя.

В соответствии с законами физики все, что выделяет большое количество тепла, должно потреблять большое количество энергии. В случае с портативным компьютером, который работает от батареи с ограниченным зарядом, потребление большого количества энергии нежелательно. Чтобы решить эту проблему,

компания Intel нашла способ переводить центральный процессор в режим пониженного энергоснабжения (состояние «сна»), если он не выполняет никаких действий, и вообще отключать его (вводить в состояние «глубокого сна»), если есть вероятность, что он не будет выполнять никаких действий некоторое время. Всего предусмотрено пять различных состояний — от полной активности до глубокого сна. В промежуточных состояниях одни функции работают (например, функция слежения кэша, обработка прерываний), другие — отключаются. В состоянии глубокого сна значения кэш-памяти и регистров сохраняются, а тактовый генератор и все внутренние блоки отключаются. Выход из «глубокого сна» происходит по специальному аппаратному сигналу. Видит ли Core i7 сны во время «глубокого сна», науке пока не известно.

Цоколевка процессора Core i7

Из 1155 контактов Core i7 для сигналов используются 447, для питания (с различным напряжением) — 286, для «земли» — 360; еще 62 зарезервированы на будущее. Для некоторых логических сигналов требуются два и более выводов (например, для запроса адреса памяти), поэтому существует только 131 вариант сигналов. Цоколевка Core i7 в несколько упрощенном виде представлена на рис. 3.42. С левой стороны рисунка показано 5 основных групп сигналов шины памяти; с правой стороны расположены прочие сигналы.

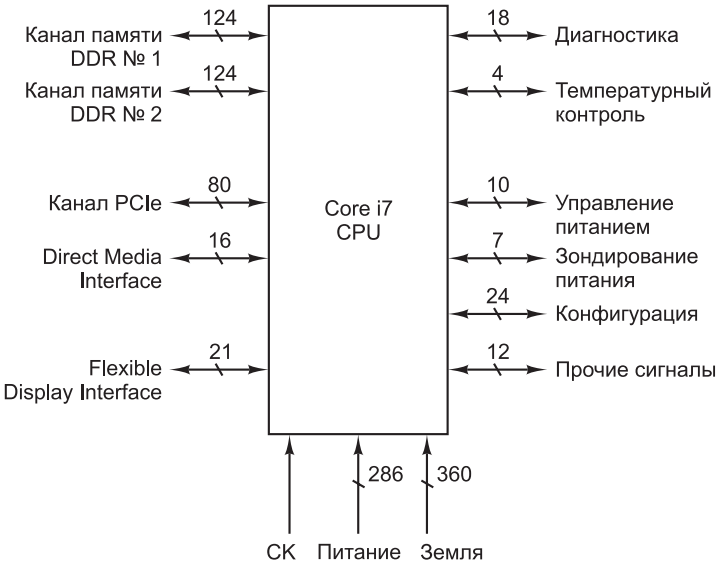


Рис. 3.42. Цоколевка процессора Core i7

Рассмотрим различные типы сигналов, начиная с сигналов шины. Первые два сигнала шины используются для взаимодействия с DDR3-совместимой динамической памятью. Группа сигналов предоставляет банку динамической памяти адрес, данные, управляющую информацию и синхронизацию. Core i7 поддерживает два независимых канала DDR3, работающих на частоте шины 666 МГц с передачей данных по фронту и по спаду сигнала; таким образом, возможно до

1333 млн взаимодействий в секунду. Интерфейс DDR3 является 64-разрядным, то есть два интерфейса DDR3 совместными усилиями ежесекундно предоставляют программам до 20 Гбайт данных.

Третья группа используется интерфейсом PCI, предназначенным для прямой связи периферийных устройств с центральным процессором Core i7. PCI Express — высокоскоростной последовательный интерфейс, в котором каждый последовательный канал образует «тракт» обмена данными с периферийными устройствами. Core i7 поддерживает интерфейс x16, то есть может одновременно использовать 16 трактов с совокупной пропускной способностью 16 Гбайт/с. Хотя канал является последовательным, через PCI Express передаются самые разнообразные команды, включая команды чтения с устройства, записи, прерывания и настройки конфигурации.

Следующая группа сигналов образует интерфейс **DMI** (Direct Media Interface), используемый для связи процессора Core i7 с комплектным чипсетом. Интерфейс DMI схож с PCI Express, хотя и работает на половине скорости последнего, поскольку четыре тракта могут обеспечить скорость передачи данных только до 2,5 Гбайт/с. Чипсет содержит полнофункциональную поддержку дополнительных периферийных интерфейсов, обычно необходимую для высокопроизводительных систем с многочисленными устройствами ввода-вывода. Чипсет Core i7 состоит из микросхем P67 и ICH10. Микросхема P67 обеспечивает поддержку интерфейсов SATA, USB, аудио, PCIe и флэш-памяти. Микросхема ICH10 обеспечивает поддержку наследных интерфейсов, включая интерфейс PCI и функциональность контроллера прерываний 8259A. Кроме того, ICH10 содержит много других схем: часы реального времени, таймеры событий и контроллеры прямого доступа к памяти (DMA). Существование таких микросхем значительно упрощает сборку полноценного персонального компьютера.

Core i7 может осуществлять прерывания тем же способом, что и 8088 (это требуется в целях совместимости), или использовать новую систему прерывания с устройством **APIC** (Advanced Programmable Interrupt Controller — **усовершенствованный программируемый контроллер прерываний**). Core i7 может действовать на любом из нескольких предустановленных напряжений, но процессор должен знать, на каком именно напряжении ему предстоит работать. Сигналы управления питанием используются для автоматического выбора напряжения источника питания, оповещения процессора о стабильности питания и ряда других родственных операций. С их же помощью осуществляется переход в различные состояния сна, которые, естественно, являются одними из инструментов управления питанием.

Несмотря на сложный механизм управления питанием, температура Core i7 может достигать очень высоких значений. Группа сигналов температурного контроля позволяет процессору оповещать окружающие устройства об опасности перегрева. Сюда относится, например, сигнал, который выдается центральным процессором, если его внутренняя температура превышает 130 °C (266 °F). Хотя если температура центрального процессора превышает 130 °C, он уже, вероятно, мечтает о выходе на пенсию и добросовестной службе в качестве нагревателя.

Впрочем, даже на таких запредельных температурах вам не придется беспокоиться о безопасности Core i7. Если внутренние датчики обнаруживают, что процессор вскоре перегреется, они запускают **терморегуляцию** — механизм, бы-

стро снижающий выделение тепла за счет того, что процессор работает только на каждом *N*-м такте. Чем выше значение *N*, тем сильнее замедляется процессор и тем быстрее он остывает. Конечно, за терморегуляцию приходится расплачиваться снижением производительности системы. До изобретения терморегуляции в случае недостаточно эффективного охлаждения процессор перегорал. Доказательства этих «мрачных времен» температурного контроля можно найти на YouTube (поищите по строке «exploding CPU»). Видеоролик поддельный, но проблема настоящая.

Сигнал Группа сигналов тактовой частоты отвечает за определение частоты системной шины. Группа диагностических сигналов предназначена для тестирования и отладки систем согласно стандарту IEEE 1149.1 JTAG. Группа сигналов инициализации обслуживает загрузку (запуск) системы.

Сигнал *CK* используется процессором для генерирования различных тактовых импульсов с частотой, кратной или дробной по отношению к частоте системного генератора. Для этого применяется устройство, называемое **системой автоподстройки по задержке**, или **DLL** (Delay-Locked Loop)

Группа диагностических сигналов предназначена для тестирования и отладки систем согласно стандарту IEEE 1149.1 JTAG (Joint Test Action Group). Наконец, в группу «прочих сигналов» отнесены разнородные сигналы, используемые для разных специальных целей.

Конвейерный режим шины памяти процессора Core i7

Современные процессоры, такие как Core i7, предъявляют жесткие требования к динамической памяти. Они работают гораздо быстрее, чем медленная динамическая память может выдавать значения, причем эта проблема усугубляется, когда несколько процессов выдают одновременные запросы. Чтобы процессор не простаивал, необходима максимально возможная производительность памяти. По этой причине шина памяти процессора Core i7 DDR3 работает в конвейерном режиме, когда в шине происходят одновременно до четырех операций. Понятие конвейера мы рассматривали в главе 2, когда говорили о конвейерных процессорах (см. рис. 2.4), но память тоже может быть конвейерной.

Чтобы конвейерный режим стал возможным, запросы к памяти Core i7 состоят из трех этапов:

1. Фаза активизации (ACT) памяти «открывает» строку динамической памяти, делая ее готовой для последующих обращений.
2. В фазе чтения (READ) или записи (WRITE) могут происходить обращения к отдельным словам открытой строки динамической памяти или к последовательным словам текущей строки динамической памяти с использованием пакетного режима.
3. Фаза предзаряда (PCHRG) «закрывает» текущую строку динамической памяти и готовит память к следующей команде активизации.

Конвейерная работа с памятью процессора Core i7 основана на том, что динамическая память DDR3 на микросхеме состоит из нескольких банков. **Банк** представляет собой блок динамической памяти, к которому процессор может обращаться параллельно с другими банками, даже находящимися на той же микросхеме. Типичная микросхема динамической памяти DDR3 содержит до восемь банков. Впрочем, спецификация интерфейса DDR3 разрешает не более

четырех параллельных обращений для одного канала DDR3. Временна́я диаграмма на рис. 3.43 показывает, как Core i7 выдает 3 обращения к трем разным банкам динамической памяти. Обращения полностью перекрываются, так что операции чтения на микросхеме динамической памяти выполняются параллельно. Связь между командами и последующими операциями на временно́й диаграмме обозначается стрелками.

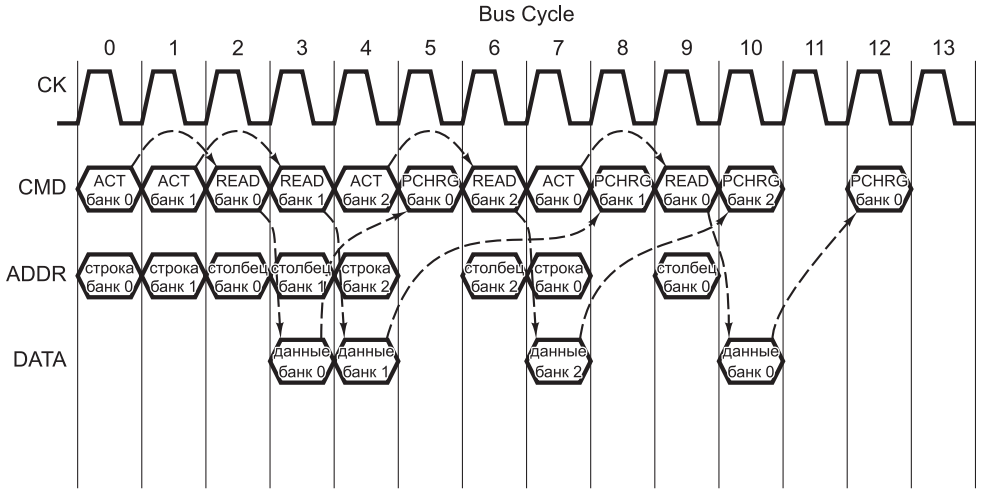


Рис. 3.43. Конвейерные обращения к памяти через интерфейс DDR3 процессора Core i7

Как видно из рис. 3.43, интерфейс памяти DDR3 имеет четыре основных сигнальных канала: синхронизация шины (CK), команда шины (CMD), адрес (ADDR) и данные (DATA). Сигнал синхронизации шины CK управляет всей работой шины. Командный сигнал CMD указывает, какая операция запрашивается у динамической памяти. Команда ACT задает адрес строки динамической памяти, открытой сигналом ADDR. При выполнении команды READ адрес столбца динамической памяти задается с использованием сигналов ADDR, а динамическая память выдает прочитанное значение спустя фиксированное время через сигналы DATA.

Наконец, команда PCHRG указывает банк, к которому применяется операция предзаряда, через сигналы ADDR. В нашем примере команда ACT должна предшествовать первой команде READ для того же банка на два цикла шины DDR3, а данные выдаются через один цикл после команды READ. Кроме того, операция PCHRG должна произойти по крайней мере на два цикла позже последней операции READ с тем же банком динамической памяти.

Параллелизм запросов памяти проявляется в перекрытии запросов READ к разным банкам динамической памяти. Первые два обращения READ к банкам 0 и 1 полностью перекрываются, производя результаты в циклах шины 3 и 4 соответственно. Обращение к банку 2 частично перекрывается с первым обращением к банку 1, и наконец, второе чтение из банка 0 частично перекрывается с обращением к банку 2.

Как Core i7 узнает, когда следует ожидать возвращения данных команды READ и когда можно выдавать новый запрос к памяти? Для этого он осущест-

вляет полное моделирование внутренней деятельности каждой подключенной микросхемы DDR3. Соответственно он ожидает возвращения данных в правильно выбранном цикле и знает, что операцию предзаряда не следует начинать раньше чем через два цикла после последней операции чтения. Core i7 может прогнозировать все эти события, потому что интерфейс памяти DDR3 работает синхронно, так что все операции занимают четко определенное количество тактов шины DDR3. Даже при наличии всей этой информации построение высокопроизводительного, полностью конвейерного интерфейса памяти DDR3 — нетривиальная задача, требующая применения многочисленных внутренних таймеров и детекторов конфликтов для реализации эффективной обработки запросов.

Однокристальная система Texas Instruments OMAP4430

В качестве второго примера процессора возьмем однокристальную систему Texas Instruments (TI) OMAP4430. OMAP4430 реализует набор команд ARM, а основной областью его применения являются мобильные и встроенные системы — смартфоны, планшетные компьютеры, интернет-гаджеты. Однокристальная система включает широкий диапазон таких устройств, чтобы при объединении ее с физической периферией (сенсорным экраном, флэш-памятью) формировалось полноценное компьютерное устройство.

OMAP4430 включает два ядра ARM A9, дополнительные ускорители и многочисленные интерфейсы периферийных устройств. Внутренняя организация OMAP4430 изображена на рис. 3.44. Ядра ARM A9 относятся к суперскалярной микроархитектуре ширины 2. Также на подложке OMAP4430 размещаются еще три процессора-ускорителя: графический процессор POWERVR SGX540, процессор обработки изображений (ISP, Image Signal Processor) и мультимедийный ускоритель IVA3.SGX540 — обеспечивает эффективную программируемую 3D-визуализацию и может рассматриваться как аналог графических процессоров для настольных компьютеров (хотя и уступающий им по скорости и мощности). ISP — программируемый процессор для эффективной обработки изображений (операции, необходимые в мощных цифровых фотокамерах). IVA3 реализует эффективное кодирование и декодирование видео с производительностью, достаточной для поддержки 3D-приложений (как, например, в ручных игровых устройствах). Также однокристальная система OMAP4430 содержит широкий спектр периферийных интерфейсов, включая сенсорные экраны и контроллеры клавиатуры, интерфейсы динамической и флэш-памяти, USB и HDMI. Фирма Texas Instruments опубликовала планы развития серии процессоров OMAP. В будущих архитектурах будет больше всего — больше ядер ARM, графических процессоров и разнообразных периферийных устройств.

Однокристальная система OMAP4430 впервые появилась в начале 2011 года. Она имела два ядра ARM A9, работавших на частоте 1 ГГц, с применением 45-нанометровой реализации. Ключевая особенность процессора OMAP4430 заключается в том, что он выполняет значительный объем вычислений с очень низкими энергозатратами, поскольку ориентируется на мобильные устройства, получающие питание от батарей. Чем эффективнее работает архитектура мобильного устройства, тем реже придется пользователю ставить устройство на зарядку.

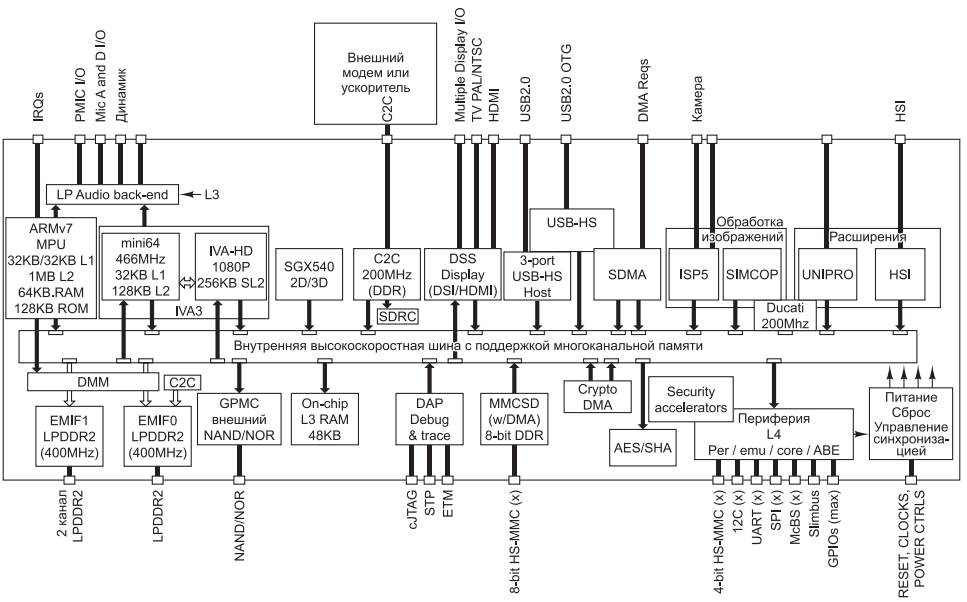


Рис. 3.44. Внутреннее строение однокристальной системы OMAP4430

Процессоры OMAP4430 выбраны с расчетом на достижение основной цели низкого энергопотребления. Графический процессор, ISP и IVA3 представляют собой программируемые ускорители, эффективно выполняющие вычисления при существенно меньших энергозатратах по сравнению с проведением тех же вычислений непосредственно на процессорах ARM A9. При полном энергопотреблении OMAP4430 использует всего 600 мВт мощности. Это составляет примерно 1/250 от энергопотребления высокопроизводительного Core i7. OMAP4430 также реализует очень эффективный спящий режим; при «засыпании» всех компонентов энергопотребление составляет всего 100 микроватт. Эффективные спящие режимы очень важны для мобильных устройств, проводящих много времени в режиме ожидания (как, например, сотовые телефоны). Чем меньше энергии расходуется в спящем режиме, тем дольше сотовый телефон сможет пребывать в ожидании.

Для дальнейшего сокращения энергопотребления в архитектуру OMAP4430 включены различные средства управления питанием, включая **динамическое масштабирование напряжения** и **ограничение питания**. Динамическое масштабирование напряжения позволяет компонентам медленнее работать на пониженном напряжении, что существенно снижает требования к питанию. Если вам не нужно, чтобы процессор выполнял вычисления на максимальной скорости, напряжение можно понизить — процессор работает медленнее, экономя значительное количество энергии. Механизм ограничения питания (power gating) использует еще более активный принцип управления питанием: неиспользуемый компонент полностью отключается и не потребляет энергии. Например, когда пользователь не смотрит видео на планшетном компьютере, видеопроцессор IVA3 полностью отключается и не потребляет энергии. И наоборот, во время просмотра IVA3 интенсивно выполняет работу по декодированию видеопотока, а два ядра ARM A9 «засыпают».

Несмотря на свою специализацию (экономия энергопотребления), ядра ARM A9 используют достаточно мощную микроархитектуру. За каждый такт они могут декодировать и выполнять до двух команд. Как мы узнаем в главе 4, эта скорость выполнения обеспечивает максимальную производительность микроархитектуры. Но не стоит полагать, что за каждый такт будут выполняться именно столько команд. Скорее, это гарантированная фирмой-изготовителем максимальная производительность; уровень, который не будет превышен процессором ни при каких условиях. Во многих тактах будет выполняться менее двух команд; это связано с наличием «препятствий», замедляющих выполнение команд и снижающих общую производительность системы. Для устранения таких ограничителей в ARM A9 встроена мощная система прогнозирования переходов, планировщик команд с изменением последовательности, и высокооптимизированная система памяти.

Система памяти OMAP4430 содержит два внутренних кэша L1 для каждого из процессоров ARM A9; 32-килобайтный кэш команд и 32-килобайтный кэш данных. Кэши обслуживаются двойными каналами с низким энергопотреблением LPDDR2. Стандарт LPDDR2 является производным от стандартного интерфейса памяти DDR2, но использует меньше проводников и работает на напряжениях, более эффективных по энергопотреблению. Кроме того, контроллер памяти содержит ряд оптимизаций (таких, как мозаичная предварительная выборка и поддержка ротации).

Хотя кэширование будет подробно рассмотрено в главе 4, сейчас о нем тоже стоит сказать пару слов. Вся основная память делится на строки кэша (блоки), состоящие из 32 байт. В кэше первого уровня хранятся 1024 наиболее интенсивно используемых строк команд и 1024 наиболее интенсивно используемых строк данных. Строки кэша, которые интенсивно используются, но не помещаются в кэше первого уровня, хранятся в кэше второго уровня. Этот кэш содержит как строки данных, так и строки команд от обоих процессоров ARM A9, смешанные произвольным образом. Кэш второго уровня содержит 32 768 строк основной памяти, к которым относились последние обращения. При промахе кэша первого уровня процессор отправляет идентификатор искомой строки кэшу второго уровня. Ответ содержит информацию, по которой процессор может определить, хранится ли указанная строка в кэше второго уровня, и если хранится — в каком состоянии. Если строка находится в кэше, процессор получает ее. Получение данных из кэша второго уровня выполняется за 19 тактов. Ожидание получается довольно долгим, поэтому квалифицированные программисты оптимизируют свою программу, чтобы она использовала меньше данных; тем самым повышается вероятность нахождения данных в быстром кэше первого уровня.

Если строка кэша отсутствует в кэше второго уровня, она загружается из основной памяти через интерфейс памяти LPDDR2. Интерфейс LPDDR2 в OMAP4430 встроен в кристалл, что делает возможным прямую связь OMAP4430 с динамической памятью. Для обращения к памяти процессор сначала отправляет микросхеме динамической памяти верхнюю часть адреса по 13 адресным линиям. Эта операция, называемая «активизацией» (ACTIVATE), загружает всю строку памяти из динамической памяти в буфер строк. Соответственно процессор может выдать несколько команд READ или WRITE, передавая остаток адреса по тем же 13 адресным линиям и отправляя (или получая) данные для операции по 32 линиям данных.

Во время ожидания результатов процессор может заниматься другой работой. Например, кэш-промах во время предварительной выборки команды не препятствует выполнению одной или нескольких уже выбранных команд, в каждой из которых могут быть задействованы данные, не находящиеся ни в одном кэше. Таким образом, даже у одного процессора может оставаться несколько незавершенных транзакций по двум интерфейсам LPDDR2. Контроллер памяти должен отслеживать текущие события и выдавать запросы к памяти в наиболее эффективном порядке.

Данные из памяти могут поступать частями по 4 байта. Операция с памятью может использовать чтение или запись в пакетном режиме, в котором происходит чтение или запись нескольких смежных адресов одной строки динамической памяти. Такой режим особенно эффективен для чтения или записи блоков кэша. Приведенное выше описание OMAP4430 (как и предшествующее ему описание Core i7) сильно упрощено, но суть происходящего в нем отражена.

Микросхема OMAP4430 содержит 547 выводов в корпусе **BGA** (рис. 3.45). Корпус BGA (Ball Grid Array) похож на LGA, но вместо квадратных площадок, используемых в LGA, в нем используются маленькие металлические шарики. Эти два типа корпусов несовместимы, что лишний раз доказывает справедливость поговорки «круглую дырку не заткнешь квадратной пробкой». Выводы OMAP4430 образуют прямоугольную матрицу размером 28 × 26, в которой отсутствуют два внутренних кольца шариков, а также две асимметричных полустроки и полустолбца для исключения возможности неправильной установки микросхемы в гнезде BGA.

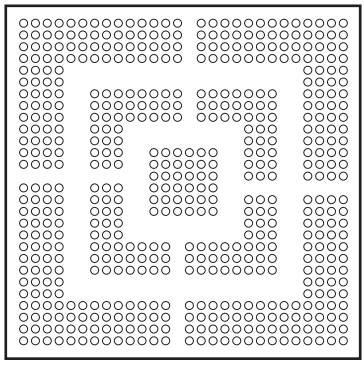


Рис. 3.45. Микросхема однокристальной системы OMAP4430

Трудно сравнивать CISC-микросхему (такую, как Core i7) с RISC-микросхемой (такой, как OMAP4430) на основании одной лишь тактовой частоты. Например, у двух ядер ARM A9 пиковая скорость выполнения достигает четырех команд на такт; в этом OMAP4430 почти сравнивается со суперскалярными процессорами ширины 4 у Core i7. Однако Core i7 быстрее выполняет программы, потому что он содержит до шести процессоров, тактовая частота которых в 3,5 раза выше (3,5 ГГц), чем у OMAP4430. Может показаться, что OMAP4430 напоминает черепаху, которая пытается обогнать зайца Core i7, однако «черепаха» расходует намного меньше энергии и первой придет к финишу — особенно если заряд батареи у «зайца» не слишком велик.

Микроконтроллер Atmel ATmega168

Core i7 и OMAP4430 — высокопроизводительные процессоры, разработанные для создания быстродействующих вычислительных устройств (Core i7 предназначен для настольных компьютеров, OMAP4430 — для мобильных систем). Однако существуют и другие компьютеры, на самом деле куда более многочисленные — так называемые встроенные системы. В этом разделе мы познакомимся с ними.

Не будет преувеличением сказать, что практически любое электронное устройство стоимостью более 100 долларов содержит встроенный компьютер. Телевизоры, сотовые телефоны, электронные секретари, микроволновые печи, видеокамеры, видеомагнитофоны, лазерные принтеры, системы охранной сигнализации, слуховые аппараты, электронные игры и многие другие устройства (их можно перечислять до бесконечности) управляются компьютером. При этом упор делается не на высокую производительность, а на низкую стоимость встроенного компьютера, что приводит к несколько другому соотношению достоинств и недостатков по сравнению с процессорами, которые мы обсуждали до сих пор.

В главе 1 мы уже упоминали о том, что в настоящее время наиболее распространенным микроконтроллером является ATmega168. Такая популярность, в первую очередь, обусловлена его низкой стоимостью. Как вы вскоре убедитесь, ATmega168 — это универсальная микросхема, к которой очень легко и недорого подключать другие устройства. Ее физическая компоновка показана на рис. 3.46.

Как видно из рисунка, ATmega168 обычно поставляется в стандартном корпусе с 28 выводами (хотя для отдельных вариантов применения предусмотрены и другие корпуса). Сразу же бросается в глаза некоторая странность этой микросхемы по сравнению с двумя предыдущими примерами: у нее нет адресных линий и линий данных. Дело в том, что микросхема не предназначена для подключения к памяти — только к устройствам. Вся память (статическая и флэш-память) содержится в самом процессоре, вследствие чего необходимость в отдельных адресных линиях и линиях данных отпадает (рис. 3.47).

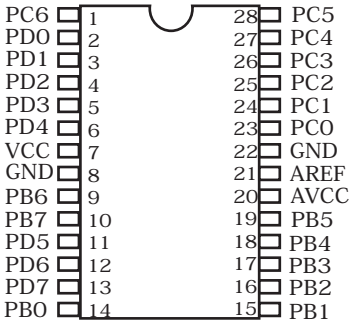


Рис. 3.46. Физическая компоновка микросхемы ATmega168

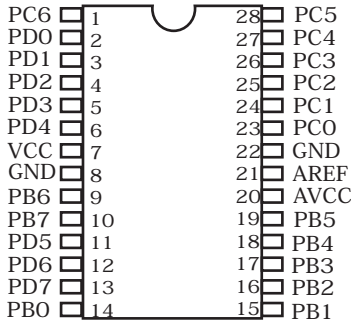


Рис. 3.47. Внутренняя архитектура и логическая компоновка ATmega168

Вместо адресных линий и линий данных ATmega168 содержит 27 портов цифрового ввода-вывода, 8 линий портов B и D и 7 линий порта C. Линии цифрового ввода-вывода предназначены для подключения периферийных устройств ввода-вывода, причем программа инициализации может задать режим работы

каждой линии (ввод или вывод). Например, при использовании в микроволновой печи одна цифровая линия может получать ввод от датчика «открытой двери», а другая — выдавать выходной сигнал для включения и выключения СВЧ-генератора. Перед включением СВЧ-генератора программное обеспечение ATmega168 проверяет, закрыта ли дверь печи. Если дверь внезапно откроется, программа должна срочно отключить питание. Впрочем, на практике также всегда применяется аппаратная блокировка.

Кроме того, шесть линий порта C могут быть настроены для аналогового ввода-вывода. Линии аналогового ввода-вывода могут читать уровень входного напряжения и задавать уровень выходного напряжения. Так, в контексте предыдущего примера у некоторых микроволновых печей имеется датчик, который позволяет пользователю разогреть еду до заданной температуры. Датчик температуры подключается к входу порта C, программа читает напряжение датчика и преобразует его в температуру с использованием функции преобразования для конкретного датчика. Остальные линии ATmega168 — вход питания (VCC), два заземления (GND) и две линии для настройки схем аналогового ввода-вывода (AREF, AVCC).

С точки зрения внутренней архитектуры ATmega168, как и OMAP4430, представляет собой однокристальную систему с широким набором внутренних устройств и памяти. ATmega168 оснащается 16 Кбайт внутренней флэш-памяти для хранения редко изменяющейся энергонезависимой информации (например, команд программы). Также имеется 1 Кбайт EEPROM — энергонезависимой памяти, запись в которую может осуществляться на программном уровне. В EEPROM хранятся данные конфигурации системы. Возвращаясь к нашему примеру с микроволновой печью: скажем, в EEPROM будет храниться бит, определяющий формат отображения времени (12- или 24-часовой). ATmega168 также содержит до 1 Кбайт внутренней статической памяти, в которой программное обеспечение может хранить временные переменные.

Во внутреннем процессоре используется набор команд AVR. Он состоит из 131 команды, каждая из которых имеет длину 16 бит. Сам процессор является 8-разрядным; это означает, что он работает с 8-разрядными значениями данных, а размер его внутренних регистров составляет 8 бит. В набор команд входят специальные команды, позволяющие 8-разрядному процессору эффективно работать с большими типами данных. Например, для выполнения 16-разрядного сложения (или сложения с большей разрядностью) процессор поддерживает команду «сложения с переносом», которая суммирует два значения и перенос от предыдущей операции. Среди остальных внутренних компонентов — часы реального времени и разнообразная интерфейсная логика: поддержка последовательных каналов, поддержка каналов широтно-импульсной модуляции, канала I2C (шина Inter-IC), контроллеры аналогового и цифрового ввода-вывода.

Примеры шин

Шины соединяют компьютерную систему в единое целое. В этом разделе мы рассмотрим популярные шины PCI и USB. Шина PCI — основная шина периферийного ввода-вывода, используемая в современных PC. Она существует

в двух разновидностях: более старая шина PCI и новая, более скоростная шина PCI Express (PCIe). USB — чрезвычайно популярная шина ввода-вывода для периферийных устройств с невысоким быстродействием (таких, как мыши и клавиатуры). Вторая и третья версии USB работают на более высоких скоростях. В следующих разделах мы поочередно рассмотрим каждую из этих шин.

Шина PCI

В первых компьютерах IBM PC большинство приложений были текстовыми. С появлением Windows постепенно вошли в употребление графические пользовательские интерфейсы. Ни одно из этих приложений особо не нагружало шину ISA. Однако с течением времени появилось множество различных приложений, в том числе игр, для которых требовалось полноэкранное видео, и ситуация коренным образом изменилась.

Давайте произведем небольшие вычисления. Рассмотрим монитор размером 1024 × 768 с 3 байтами на пиксел. Одно экранное изображение содержит 2,25 Мбайт данных. Для воспроизведения плавных движений требуется 30 кадров в секунду, и, следовательно, скорость передачи данных должна быть 67,5 Мбайт/с. В действительности дело обстоит гораздо хуже, поскольку, чтобы передать изображение, данные нужно передать с жесткого диска, компакт-диска или DVD-диска через шину в память. Затем данные должны поступить в графический адаптер (тоже через шину). Таким образом, только для передачи видео пропускная способность шины должна быть 135 Мбайт/с, без учета потребностей центрального процессора и других устройств.

Максимальная частота передачи данных предшественницы PCI — шины ISA — составляла 8,33 МГц. Шина ISA способна передавать два байта за цикл, поэтому ее максимальная пропускная способность составляет 16,7 Мбайт/с. Шина EISA может передавать 4 байта за цикл. Ее пропускная способность достигает 33,3 Мбайт/с. Ясно, что ни одна из них совершенно не соответствует тому, что требуется для полноэкранного видео.

С современным видео формата Full HD дело обстоит еще хуже. Для него необходимо воспроизведение 1920 × 1080 с частотой 30 кадров в секунду, поэтому скорость передачи данных должна составлять 155 Мбайт/с (или 310 Мбайт/с, если данные должны проходить по шине дважды). Разумеется, шина EISA даже близко не соответствовала этим требованиям.

В 1990 году компания Intel разработала новую шину с гораздо более высокой пропускной способностью, чем у шины EISA. Эту шину назвали **PCI** (Peripheral Component Interconnect — **взаимодействие периферийных компонентов**). Компания Intel запатентовала шину PCI и сделала все патенты всеобщим достоянием, так что любая компания могла производить периферийные устройства для этой шины без каких-либо выплат за право пользования патентом. Компания Intel также сформировала промышленный консорциум PCI Special Interest Group, который должен был заниматься дальнейшими усовершенствованиями шины PCI. Все эти действия привели к тому, что шина PCI стала чрезвычайно популярной. Фактически в каждом компьютере Intel (начиная с Pentium), а также во многих других компьютерах есть шина PCI. Шина PCI подробно описана в литературе [Shanley and Anderson, 1999; Solari and Willse, 2004].

Первая шина PCI передавала 32 бита за цикл и работала на частоте 33 МГц (время цикла — 30 нс), общая пропускная способность составляла 133 Мбайт/с. В 1993 году появилась шина PCI 2.0, а в 1995 году — PCI 2.1. Шина PCI 2.2 подходит и для портативных компьютеров (где требуется экономии заряда батареи). В конце концов удалось получить шину PCI, которая работает на частоте 66 МГц, способна передавать 64 бита за цикл, а ее общая пропускная способность составляет 528 Мбайт/с. При такой производительности полноэкранное видео вполне достижимо (если диск и другие устройства системы справляются со своей частью работы). Во всяком случае, шина PCI не является «узким местом» системы.

Хотя 528 Мбайт/с — достаточно высокая скорость передачи данных, все же здесь есть некоторые проблемы. Во-первых, этого недостаточно для шины памяти. Во-вторых, шина PCI несовместима со всеми старыми платами ISA. По этой причине компания Intel решила разрабатывать компьютеры с тремя и более шинами, как показано на рис. 3.48. Здесь мы видим, что центральный процессор может обмениваться информацией с основной памятью через специальную шину памяти, а шину ISA можно связать с шиной PCI. Такая архитектура в 1990-х годах удовлетворяла всем современным на тот момент требованиям и поэтому использовалась в большинстве систем.

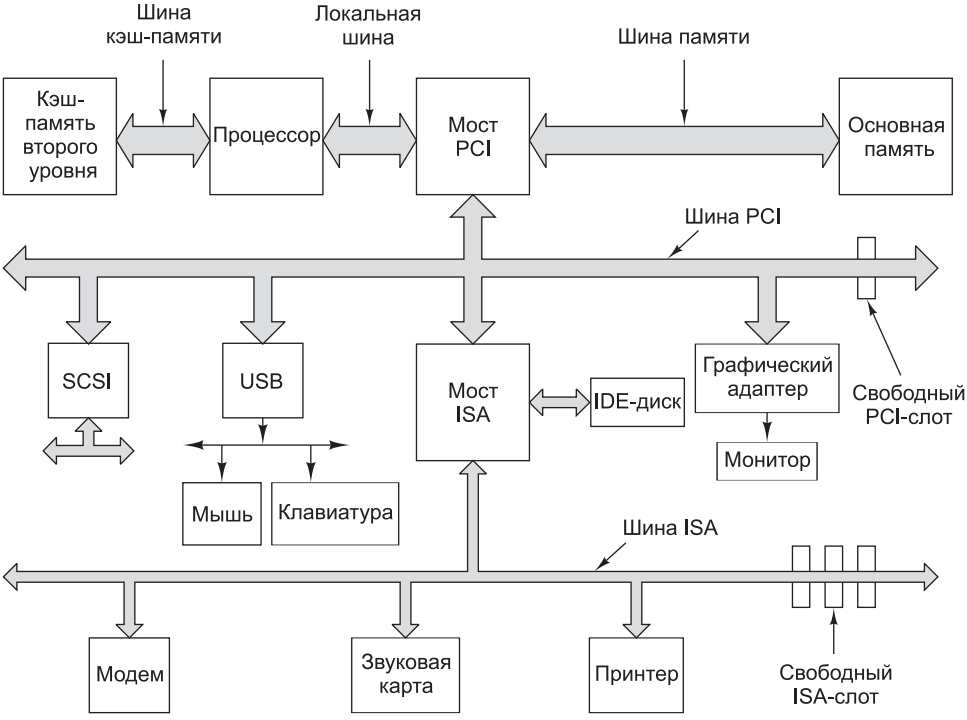


Рис. 3.48. Архитектура типичной системы первых поколений Pentium. (Толщина линий шины обозначает ее пропускную способность. Чем толще линия, тем выше пропускная способность)

Ключевыми компонентами данной архитектуры являются мосты между шинами (эти микросхемы выпускает компания Intel — отсюда такой интерес

к проекту). Мост PCI связывает центральный процессор, память и шину PCI. Мост ISA связывает шину PCI с шиной ISA, а также поддерживает один или два IDE-диска. Практически все PC, использующие эту архитектуру, выпускаются с одним или несколькими свободными PCI-слотами для подключения дополнительных высокоскоростных периферийных устройств и с одним или несколькими ISA-слотами для подключения низкоскоростных периферийных устройств.

Преимущество системы, изображенной на рис. 3.48, состоит в том, что шина между центральным процессором и памятью имеет чрезвычайно высокую пропускную способность, шина PCI также обладает высокой пропускной способностью и хорошо подходит для взаимодействия с высокоскоростными периферийными устройствами (SCSI-дисками, графическими адаптерами и т. п.), и при этом еще могут использоваться старые платы ISA. На рисунке также изображена шина USB, которую мы будем обсуждать далее в этой главе.

Было бы неплохо, если бы существовал только один тип плат PCI. К сожалению, это не так. Платы отличаются потребляемой мощностью, разрядностью и синхронизацией. Старые компьютеры обычно используют напряжение 5 В, а новые — 3,3 В, поэтому шина PCI поддерживает то и другое. Коннекторы одни и те же (они отличаются только двумя небольшими пластмассовыми вставками, не позволяющими вставить плату на 5 В в шину PCI на 3,3 В, и наоборот). К счастью, существуют и универсальные платы, которые поддерживают оба напряжения и которые можно вставить в любой слот. Платы отличаются не только напряжением, но и разрядностью. Существует два типа плат: 32-разрядные и 64-разрядные. 32-разрядные платы содержат 120 выводов; 64-разрядные платы содержат те же 120 выводов плюс 64 дополнительных вывода. Шина PCI, поддерживающая 64-разрядные платы, может поддерживать и 32-разрядные, но обратное не верно. Наконец, шины PCI и соответствующие платы могут работать на частоте либо 33 МГц, либо 66 МГц. В обоих случаях контакты идентичны. Отличие состоит в том, что один из выводов связывается либо с источником питания, либо с землей.

К концу 1990-х годов шина ISA была окончательно похоронена участниками рынка, и в новых системах ее поддержка уже не предусматривалась. В связи с тем, что разрешение экрана постоянно увеличивалось (достигнув величины 1600 × 1200 точек), равно как и спрос на полноэкранное видео со стандартной частотой кадров, особо актуальное в интерактивных играх, компания Intel разработала новую шину, предназначенную исключительно для обмена данными с графическим адаптером. Эта шина называется **AGP** (Accelerated Graphics Port — **ускоренный графический порт**). Ее первая версия, AGP 1.0, работала на скорости 264 Мбайт/с, и эта величина была принята за 1x. Недостаток скорости (по сравнению с PCI) компенсировался узкой специализацией на управлении графическим адаптером. Впоследствии были разработаны новые версии шины — в частности, AGP 3.0 работает на скорости 2,1 Гбайт/с (8x). Сегодня даже высокопроизводительная шина AGP 3.0 уступает более быстрым новинкам — прежде всего, PCI Express, способной передавать данные со скоростью до 16 Гбайт/с по высокоскоростному последовательному каналу. Схема современной системы на базе Core i7 показана на рис. 3.49.

В современной системе на базе Core i7 ряд интерфейсов встраивается прямо в микросхему процессора. Два канала памяти DDR3, работающих со скоростью

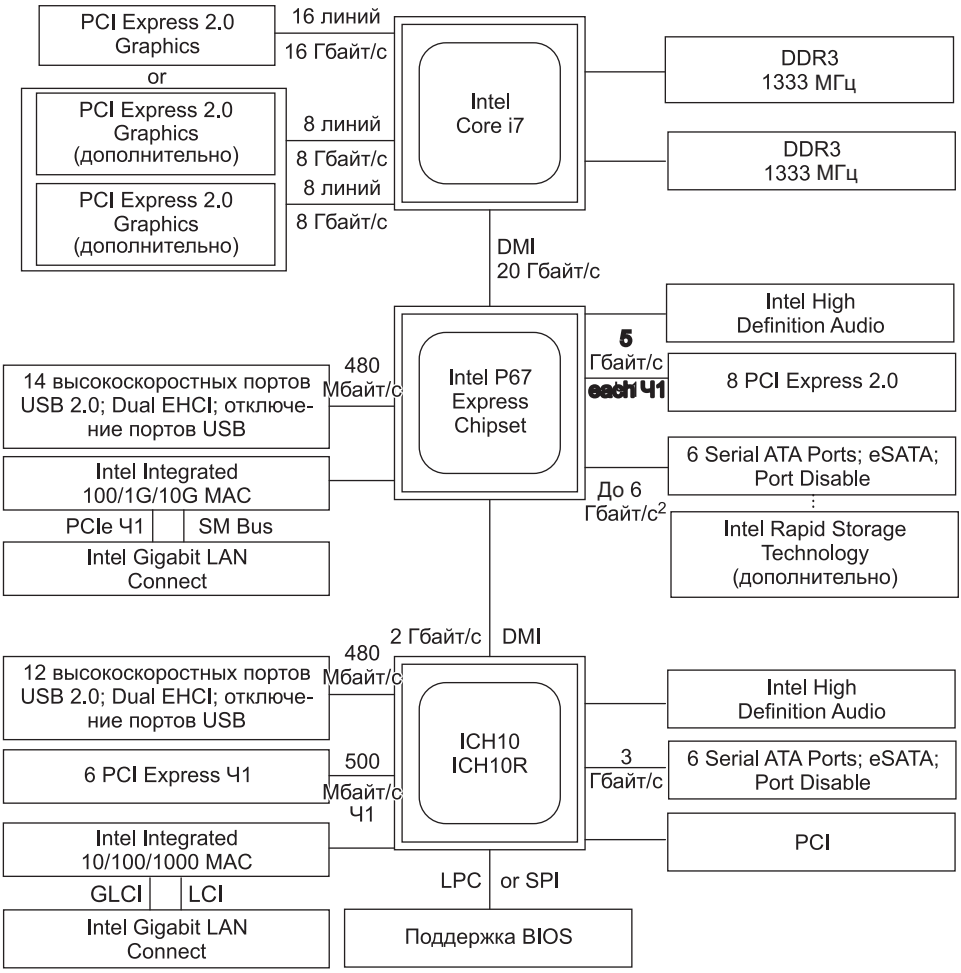


Рис. 3.49. Структура шин в современной системе Core i7

1333 транзакции/с, соединяются с основной памятью и обеспечивают суммарную пропускную способность 10 Гбайт/с на канал. Также в процессор интегрируется канал PCI Express на 16 линий, который может быть настроен для работы в режиме одной 16-разрядной шины PCI Express или двух независимых 8-разрядных шин PCI Express. 16 линий совместно обеспечивают для устройств ввода-вывода пропускную способность в 16 Гбайт/с.

Процессор соединен с основной мостовой микросхемой P67 через последовательный интерфейс DMI (Direct Media Interface) со скоростью 20 Гбит/с (2 Гбайт/с). P67 предоставляет интерфейс к нескольким современным высокопроизводительным интерфейсам ввода-вывода. Поддерживаются 8 дополнительных линий PCI Express и дисковые интерфейсы SATA. P67 также реализует 15 интерфейсов USB 2.0, 10G Ethernet и аудиоинтерфейс.

Микросхема ICH10 обеспечивает поддержку интерфейсов старых устройств. Она соединяется с P67 через медленный интерфейс DMI. ICH10 реализует шину

PCI, 1G Ethernet, порты USB ports и старые версии PCI Express и SATA. В новых системах ICH10 микросхема может отсутствовать; она необходима только в том случае, если система должна поддерживать старые интерфейсы.

Работа шины PCI

Шины PCI являются синхронными, как и все шины PC, восходящие к первой модели IBM PC. Все транзакции в шине PCI осуществляются между **задающим** и **подчиненным** устройствами. Чтобы не увеличивать число выводов на плате, адресные и информационные линии объединяются. При этом достаточно 64-х выводов для всей совокупности адресных и информационных сигналов, даже если PCI работает с 64-разрядными адресами и 64-разрядными данными.

Объединенные адресные и информационные выводы функционируют следующим образом. При считывании во время первого цикла задающее устройство передает адрес на шину. Во время второго цикла задающее устройство удаляет адрес, и шина переключается таким образом, чтобы подчиненное устройство могло ее использовать. Во время третьего цикла подчиненное устройство выдает запрашиваемые данные. При записи шине не нужно переключаться, поскольку задающее устройство передает в нее и адрес, и данные. Тем не менее минимальная транзакция занимает три цикла. Если подчиненное устройство не может дать ответ в течение трех циклов, то вводится режим ожидания. Допускаются пересылки блоков неограниченного размера, а также некоторые другие типы циклов шины.

Арбитраж шины PCI

Чтобы использовать шину PCI, устройство сначала должно получить к ней доступ. Шина PCI управляется централизованным арбитром, как показано на рис. 3.50. В большинстве случаев арбитр шины встраивается в один из мостов между шинами. От каждого PCI-устройства к арбитру тянутся две специальные линии. Одна из них (REQ#) используется для запроса шины, а вторая (GNT#) — для получения разрешения на доступ к шине. (Примечание. REQ# — обозначение $\overline{\text{REQ}}$ в терминологии PCI.)

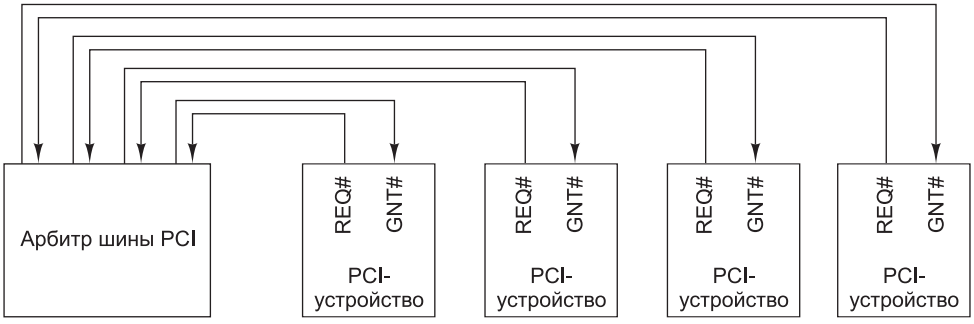


Рис. 3.50. У шины PCI имеется централизованный арбитр

Чтобы сделать запрос на доступ к шине, PCI-устройство (в том числе и центральный процессор) устанавливает сигнал REQ# и ждет, пока арбитр не установит сигнал GNT#. Если арбитр установил сигнал GNT#, то устройство может использовать шину в следующем цикле. Алгоритм, которым руководствуется

арбитр, не определяется в спецификации шины PCI. Допустимы циклический арбитраж, приоритетный арбитраж, а также другие схемы арбитража. Разумеется, хороший арбитр должен быть справедлив, чтобы не заставлять отдельные устройства ждать целую вечность.

Шина предоставляется для одной транзакции, хотя продолжительность этой транзакции теоретически не ограничена. Если устройству нужно совершить вторую транзакцию и ни одно другое устройство не запрашивает шину, оно может занять шину снова, хотя обычно между транзакциями требуется вставлять пустой цикл. Однако при особых обстоятельствах (при отсутствии конкуренции на доступ к шине) устройство может совершать последовательные транзакции без пустых циклов между ними. Если задающее устройство выполняет очень длительную передачу, а какое-нибудь другое устройство выдало запрос на доступ к шине, арбитр может сбросить сигнал на линии GNT#. Предполагается, что задающее устройство следит за линией GNT#, и при сбросе сигнала устройство должно освободить шину в следующем цикле. Такая система позволяет осуществлять очень длинные передачи (что весьма рационально) при отсутствии конкуренции на доступ к шине, однако при этом она быстро реагирует на запросы шины, поступающие от других устройств.

Сигналы шины PCI

Шина PCI поддерживает ряд обязательных (табл. 3.4) и ряд дополнительных сигналов (табл. 3.5). Оставшиеся выводы используются для питания, земли и разнообразных сопутствующих сигналов. В столбцах «Задающее устройство» и «Подчиненное устройство» указывается, какое из устройств устанавливает сигнал при обычной транзакции. Если сигнал устанавливается другим устройством (например, CLK), оба столбца остаются пустыми.

Таблица 3.4. Обязательные сигналы шины PCI

Сигнал	Количество линий	Задающее устройство	Подчиненное устройство	Комментарий
CLK	1			Тактовый генератор (33 МГц или 66 МГц)
AD	32	Да	Да	Объединенные адресные и информационные линии
PAR	1	Да		Бит четности для адреса или данных
C/BE#	4	Да		Во-первых, команда шине, во-вторых, битовый массив, который показывает, какие байты из слова нужно считать (или записать)
FRAME#	1	Да		Указывает, что установлены сигналы AD и C/BE
IRDY#	1	Да		При чтении означает, что задающее устройство готово принять данные; при записи — что данные находятся в шине

продолжение ⇩

Таблица 3.4 *(продолжение)*

Сигнал	Количество линий	Задающее устройство	Подчиненное устройство	Комментарий
IDSEL	1	Да		Считывание конфигурационного пространства
DEVSEL#	1		Да	Подчиненное устройство распознало свой адрес и ждет сигнала
TRDY#	1		Да	При чтении означает, что данные находятся на линиях AD; при записи — что подчиненное устройство готово принять данные
STOP#	1		Да	Подчиненное устройство требует немедленно прервать текущую транзакцию
PERR#	1			Обнаружена ошибка четности данных
SERR#	1			Обнаружена ошибка четности адреса или системная ошибка
REQ#	1			Арбитраж шины — запрос на доступ к шине
GNT#	1			Арбитраж шины — предоставление шины
RST#	1			Перезагрузка системы и всех устройств

Таблица 3.5. Дополнительные сигналы шины PCI

Сигнал	Количество линий	Задающее устройство	Подчиненное устройство	Комментарий
REQ64#	1	Да		Запрос на осуществление 64-разрядной транзакции
ACK64#	1		Да	Разрешение 64-разрядной транзакции
AD	32	Да		Дополнительные 32 бита адреса или данных
PAR64	1	Да		Проверка четности для дополнительных 32 бит адреса или данных
C/BE#	4	Да		Дополнительные 4 бита для указания, какие байты из слова нужно считать (или записать)
LOCK	1	Да		В многопроцессорных системах блокировка шины при осуществлении транзакции одним из процессоров

Сигнал	Количество линий	Задающее устройство	Подчиненное устройство	Комментарий
SBO#	1			В многопроцессорных системах обращение к кэш-памяти другого процессора
SDONE	1			В многопроцессорных системах сигнал о завершении слежения
INTx	4			Запрос прерывания
JTAG	5			Сигналы тестирования IEEE 1149.1 JTAG
M66EN	1			Сигнал связывается с источником питания или с землей (66 МГц или 33 МГц)

Теперь давайте рассмотрим каждый сигнал шины PCI отдельно. Начнем с обязательных (32-разрядных) сигналов, а затем перейдем к дополнительным (64-разрядным). Сигнал CLK управляет шиной. Большинство сигналов синхронизируется с ним. У шины PCI транзакция начинается на спаде сигнала CLK, то есть не в начале цикла, а в середине.

Сигналы AD (их 32) нужны для адресов и данных (для передач по 32 бита). Обычно адрес устанавливается во время первого цикла, а данные — во время третьего. Сигнал PAR — это бит четности для сигнала AD. Сигнал C/BE# выполняет две функции. Во время первого цикла он содержит команду (считать одно слово, считать блок и т. п.). Во время второго цикла он содержит массив размером 4 бита, который показывает, какие байты 32-разрядного слова действительны. Используя сигнал C/BE#, можно считать 1, 2 или 3 байта из слова, а также все слово целиком.

Сигнал FRAME# устанавливается задающим устройством, чтобы начать транзакцию. Этот сигнал сообщает подчиненному устройству, что адрес и команды в данный момент действительны. При чтении одновременно с сигналом FRAME# устанавливается сигнал IRDY#. Он сообщает, что задающее устройство готово принять данные. При записи сигнал IRDY# устанавливается позже, когда данные уже переданы в шину.

Сигнал IDSEL связан с тем, что у каждого устройства PCI должно быть конфигурационное пространство на 256 байт, которое другие устройства могут считывать (установив сигнал IDSEL). Это конфигурационное пространство содержит характеристики устройства. В некоторых операционных системах механизм автоматического конфигурирования (Plug and Play, PnP) использует это пространство, чтобы выяснить, какие устройства подключены к шине.

А теперь рассмотрим сигналы, которые устанавливаются подчиненным устройством. Сигнал DEVSEL# означает, что подчиненное устройство распознало свой адрес на линиях AD и готово участвовать в транзакции. Если сигнал DEVSEL# не поступает в течение определенного промежутка времени, задающее устройство предполагает, что подчиненное устройство, к которому направлено обращение, либо отсутствует, либо неисправно.

Следующий сигнал — TRDY#. Его подчиненное устройство устанавливает при чтении, чтобы сообщить, что данные находятся на линиях AD, и при записи, чтобы сообщить, что оно готово принять данные.

Следующие три сигнала используются для передачи сообщений об ошибках. Один из них, сигнал STOP#, устанавливается подчиненным устройством, если произошла какая-нибудь неполадка и нужно прервать текущую транзакцию. Следующий сигнал, PERR#, используется для сообщения об ошибке четности в данных на предыдущем цикле. Для чтения этот сигнал устанавливается задающим устройством, для записи — подчиненным устройством. Необходимые действия должно предпринимать устройство, получившее этот сигнал. Наконец, сигнал SERR# служит для сообщения об адресных и системных ошибках.

Сигналы REQ# и GNT# предназначены для арбитража шины. Они устанавливаются не тем устройством, которое является задающим в данный момент, а тем, которое желает стать задающим. Последний обязательный сигнал, RST#, применяется для перезагрузки системы, которая происходит либо при нажатии пользователем кнопки RESET, либо если какое-нибудь системное устройство обнаруживает фатальную ошибку. После установки этого сигнала компьютер перезагружается.

Перейдем к дополнительным сигналам, большинство из которых связано с расширением разрядности с 32 до 64 бит. Сигналы REQ64# и ACK 64# позволяют задающему устройству попросить разрешение осуществить 64-разрядную транзакцию, а подчиненному устройству принять эту транзакцию. Сигналы AD, PAR64 и C/BE# являются расширениями соответствующих 32-разрядных сигналов.

Следующие три сигнала не связаны с переходом с 32 на 64 бита. Они используются в многопроцессорных системах. Не все платы PCI поддерживают такие системы, поэтому эти сигналы отнесены к дополнительным. Сигнал LOCK позволяет блокировать шину для параллельных транзакций. Следующие два сигнала связаны с фазой слежения, позволяющей сохранить согласованность кэшей разных процессоров.

Сигналы INT*x* нужны для запроса прерываний. Плата PCI может содержать до четырех логических устройств, каждое из которых имеет собственную линию запроса прерываний. Сигналы JTAG предназначены для процедуры тестирования IEEE 1149.1 JTAG. Наконец, сигнал M66EN связывается либо с источником питания, либо с землей, что определяет тактовую частоту. Она не должна меняться во время работы системы.

Транзакции на шине PCI

Шина PCI в действительности очень проста (для современной шины, конечно). Чтобы лучше понять это, рассмотрим временну́ю диаграмму на рис. 3.51. Здесь мы видим транзакцию чтения, за ней следуют пустой цикл и транзакция записи, которая осуществляется тем же задающим устройством.

Во время цикла T_1 на спаде синхронизирующего сигнала задающее устройство помещает адрес на линии AD и команду на линии C/BE#. Затем задающее устройство устанавливает сигнал FRAME#, чтобы начать транзакцию.

Во время цикла T_2 задающее устройство переключает шину, чтобы подчиненное устройство могло воспользоваться ею во время цикла T_3. Задающее устройство также изменяет сигнал C/BE#, чтобы указать, какие байты в слове ему нужно считать.

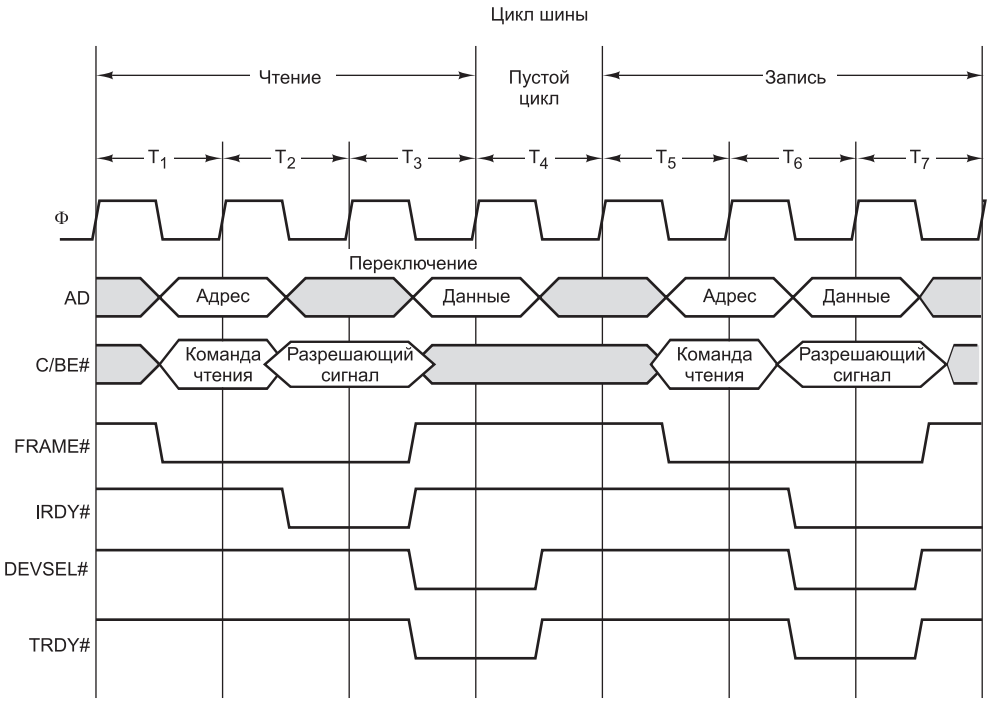


Рис. 3.51. Примеры 32-разрядных транзакций на шине PCI. Во время первых трех циклов происходит операция чтения, затем идет пустой цикл, а следующие три цикла — операция записи

Во время цикла T_3 подчиненное устройство устанавливает сигнал DEVSEL#. Этот сигнал сообщает задающему устройству, что подчиненное устройство получило адрес и собирается ответить. Подчиненное устройство также помещает данные на линии AD и выдает сигнал TRDY#, который сообщает задающему устройству о данном действии. Если подчиненное устройство не может ответить быстро, оно не снимает сигнал DEVSEL#, извещающий о присутствии этого устройства, но при этом не устанавливает сигнал TRDY# до тех пор, пока не сможет передать данные. При такой процедуре вводится один или несколько периодов ожидания.

В нашем примере (а также часто на практике) следующий цикл — пустой. Мы видим, что в цикле T_5 то же самое задающее устройство инициирует процесс записи. Сначала оно как обычно помещает адрес и команду на шину. В следующем цикле оно выдает данные. Поскольку линиями AD управляет одно и то же устройство, цикл переключения не требуется. В цикле T_7 память принимает данные.

PCI Express

Возможностей шины PCI вполне достаточно для большинства современных приложений, однако потребность в ускорении ввода-вывода постепенно усложняет некогда стройную внутреннюю архитектуру ПК. Рисунок 3.50 наглядно

свидетельствует о том, что шина PCI более не является центральным элементом, сводящим воедино компоненты ПК. Эту роль теперь исполняет мост.

Суть проблемы заключается в том, что со временем появляется все больше устройств ввода-вывода, слишком быстрых для шины PCI. Разгон тактовой частоты шины — далеко не лучшее решение, поскольку только усугубляет проблемы с расфазировкой шины, перекрестными помехами между проводниками и емкостным сопротивлением. При появлении каждого нового устройства, которое оказывается слишком быстрым для шины PCI (будь то графический адаптер, жесткий диск, сетевой контроллер и т. д.), разработчикам Intel приходится создавать очередной специализированный порт, с помощью которого мост позволяет этому устройству обходить шину PCI. Естественно, такое решение не рассчитано на долгосрочную перспективу.

Еще один недостаток шины PCI состоит в чрезмерных габаритах плат. Стандартные платы PCI обычно имеют размеры 17,5 × 10,7 см, а компактные платы — 12,5 × 3,6 см. Ни один из вариантов не умещается в корпусе современных портативных компьютеров, не говоря уже карманных моделях. В то же время производители постоянно уменьшают размеры выпускаемых устройств. Кроме того, некоторые производители планируют перейти на новую схему размещения устройств в корпусах ПК — а именно размещать процессор и память в отдельном закрытом отсеке, а жесткий диск — внутри монитора. С платами PCI такое решение не реализуемо.

Сейчас предлагается несколько вариантов решения указанных проблем, но, скорее всего, победителем в конкурентной борьбе окажется технология **PCI Express**, которую активно продвигает Intel. Несмотря на название, она не имеет почти ничего общего с шиной PCI; более того — это вообще не шина. Тем не менее маркетологи решили не избавляться от названия «PCI» — благо, оно у всех на слуху. Сейчас наличие этой шины у компьютера стало уже стандартом. Посмотрим, что она собой представляет.

Архитектура PCI Express

Суть технологии PCI Express заключается в замене параллельной шины с ее многообразием задающих и подчиненных устройств высокоскоростными двухточечными последовательными соединениями. Это решение знаменует собой окончательный отход от шинной топологии, реализованной в шинах ISA/EISA/PCI, и переход на топологию локальных сетей (особенно коммутируемых сетей Ethernet). Основная идея такова: по сути, ПК — это набор микросхем процессора, памяти и устройств ввода-вывода, которые необходимо соединить между собой. Учитывая это обстоятельство, PCI Express исполняет роль универсального коммутатора, соединяющего микросхемы по последовательным каналам. Стандартная конфигурация PCI Express изображена на рис. 3.52.

Как видно из рисунка, процессор, память и кэш подключены к мосту традиционным способом. Новым элементом здесь является подключенный к мосту коммутатор (иногда он встраивается непосредственно в микросхему моста). Между каждой микросхемой устройства ввода-вывода, с одной стороны, и коммутатором, с другой, устанавливается двухточечное соединение. Любое такое соединение состоит из двух однонаправленных каналов — по одному в каждом из направлений между устройством и коммутатором. Каналы состоят из двух про-

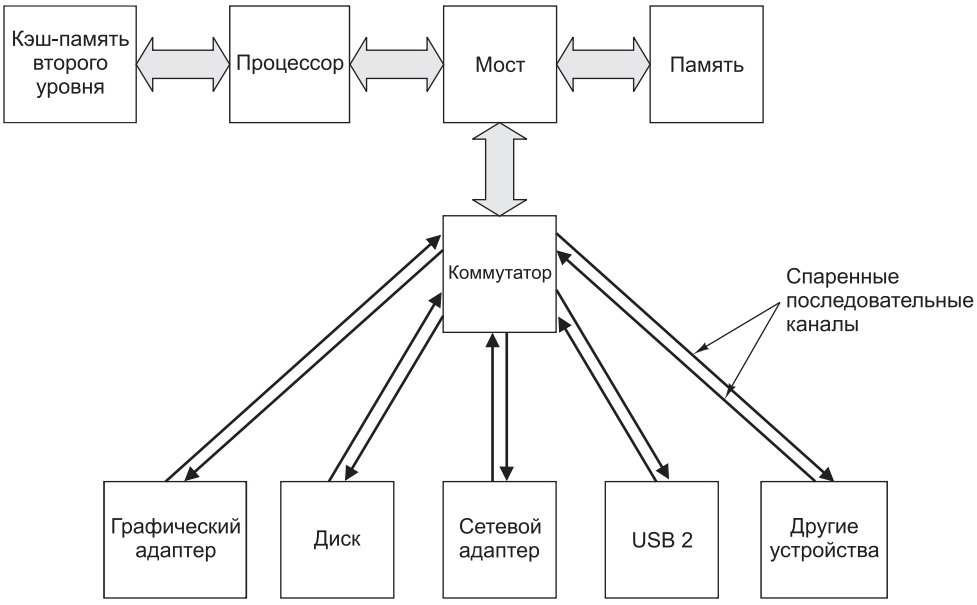


Рис. 3.52. Стандартная компоновка системы PCI Express

водов (сигнального и заземляющего), что обеспечивает высокую помехозащищенность в ходе высокоскоростной передачи сигналов. Такая архитектура отличается от предыдущей большей унификацией и равноправием всех устройств.

Три основные момента отличают архитектуру PCI Express от архитектуры PCI. Два из них мы уже рассмотрели — это наличие централизованного коммутатора, пришедшего на смену принципу многоотводной шины, и применение узких последовательных двухточечных соединений вместо широкой параллельной шины. Третье отличие не столь очевидно. Концептуальная модель, на которой основана шина PCI, сводится к тому, что задающее устройство шины передает подчиненным устройствам команды на чтение слова или блока из нескольких слов. PCI Express основывается на другом принципе, предусматривающем отправку пакетов данных от одного устройства другому. Понятие **пакета**, состоящего из заголовка и полезной нагрузки, заимствовано из сетевых технологий. В **заголовке** содержится управляющая информация, а значит, отпадает потребность в многочисленных управляющих сигналах, которые играют важную роль при передаче по шине PCI. **Полезная нагрузка** содержит непосредственно передаваемые данные. Таким образом, ПК, поддерживающий технологию PCI Express, напоминает миниатюрную сеть с коммутацией пакетов.

Помимо вышеперечисленных наиболее важных изменений есть и менее заметные. В частности, в пакетах использован более надежный по сравнению с PCI код обнаружения ошибок. Далее, физическая длина соединения между микросхемой и коммутатором увеличилась до 50 см, за счет чего стало удобнее менять положение компонентов системы. Поскольку к базовому коммутатору можно подключить другой коммутатор, сформировав, таким образом, древовидную структуру, повышается степень расширяемости системы. Кроме того, устройства в рамках PCI

Express поддерживают «горячее» подключение, а значит, их можно снимать и монтировать непосредственно в процессе работы. Наконец, так как последовательные коннекторы значительно меньше старых PCI-коннекторов, ничто не мешает производителям разрабатывать компактные устройства и компьютеры. Таким образом, PCI Express решительно отходит от принципов работы шины PCI.

Стек протоколов PCI Express

Следуя модели сети с коммутацией пакетов, технология PCI Express реализуется на основе многоуровневого стека протоколов. **Протоколом** называется набор правил, определяющих механизм взаимодействия между двумя сторонами. Соответственно, стек протоколов — это иерархическая система протоколов, которые регламентируют различные аспекты взаимодействия на тех или иных уровнях. Рассмотрим для примера деловое письмо. Существуют определенные правила относительно местоположения и содержания шапки письма, адреса получателя, даты, формы приветствия, тела письма, подписей и т. д. Все эти условности можно обобщенно назвать протоколом делового письма. Помимо этого, есть стандарты, касающиеся размера и формата конверта, размещения штампа и тому подобных тонкостей. Эти два уровня и соответствующие протоколы независимы друг от друга. К примеру, можно полностью изменить формат письма, положив его в стандартный конверт, и наоборот. Подобные многоуровневые протоколы, которые делают возможной модульную разработку с высоким уровнем гибкости, уже несколько десятилетий широко применяются в области сетевого ПО. В технологии PCI Express сделана попытка реализовать их в аппаратном обеспечении «шины».

Стек протоколов PCI Express изображен на рис. 3.53, *а*.

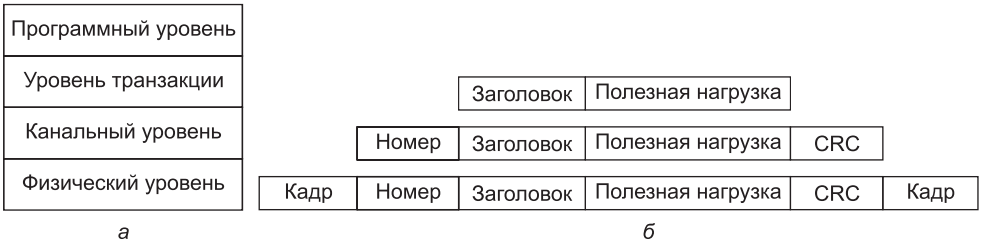


Рис. 3.53. Стек протоколов PCI Express (*а*); формат пакета (*б*)

Рассмотрим уровни по восходящей. Самым нижним является **физический уровень**. Он отвечает за передачу битов от отправителя к получателю по двухточечному соединению. Каждое такое соединение состоит из одной или нескольких пар симплексных (однонаправленных) каналов. В простейшем случае на каждое направление выделяется по одной паре, но также допустимо наличие 2, 4, 8, 16 или 32 пар. Каналы, входящие в пары, называются **полосами** (lanes). На каждое направление должно быть выделено одинаковое количество полос. В первом поколении предусматривается скорость передачи данных от 2,5 Гбайт/с для каждого направления, но через некоторое время эта цифра, вероятно, дойдет до 10 Гбайт/с.

В отличие от шин ISA, EISA и PCI, в технологии PCI Express не предусмотрен тактовый генератор. Устройства вправе начинать передачу в любой момент,

как только им будет, что передавать. Такая свобода, с одной стороны, повышает быстродействие, с другой порождает проблему. Предположим, что 1 кодируется напряжением +3 В, а 0 — напряжением 0 В. Если первые несколько байтов равны нулю, как получатель узнает о том, что ему передаются данные? Действительно — последовательность нулевых битов трудно отличить от простоя канала. Эта проблема решается при помощи так называемой **8/10-разрядной кодировки**. Согласно этой схеме, 1 байт фактических данных кодируется при помощи 10-разрядного символа. Из 1024 возможных 10-разрядных символов выбираются такие, которые за счет достаточного количества фронтов без задающего генератора обеспечивают синхронизацию отправителя и получателя по границам битов. В силу применения 8/10-разрядной кодировки суммарная пропускная способность канала, равная 2,5 Гбайт/с, сужается до фактической пропускной способности 2 Гбайт/с.

Канальный уровень отвечает за передачу пакетов. На этом уровне к заголовку и полезной нагрузке, переданным с уровня транзакций, добавляется порядковый номер и код исправления ошибок — так называемый **CRC** (Cyclic Redundancy Check — **циклический контроль избыточности**). CRC-код генерируется путем применения определенного алгоритма к заголовку и полезной нагрузке. По получении пакета устройство проводит аналогичные вычисления с заголовком и данными и сравнивает результат с величиной, указанной в пакете. Если два результата совпадают, первоначальному отправителю отсылается **пакет подтверждения** правильности полученных данных. В противном случае получатель делает запрос на повторную передачу. Таким образом, значительно повышаются показатели целостности данных по сравнению с шиной PCI, в которой не реализованы средства контроля и повторной передачи данных.

Во избежание перегрузки медленного получателя пакетами, исходящими от быстрого отправителя, реализуется механизм **управления потоками**. Этот механизм основывается на выдаче получателем отправителю определенного количества разрешений на передачу пакетов — в зависимости от объема свободного пространства, необходимого для их хранения. Исчерпав ранее выданные разрешения, отправитель должен приостановить передачу и дождаться новых разрешений. Такая схема, распространенная во всех сетях, предотвращает потерю данных вследствие несовпадения скоростей отправителя и получателя.

На **уровне транзакций** выполняются все операции шины. К примеру, для считывания слова из памяти нужно выполнить две транзакции, из которых одну инициирует процессор или канал DMA, запрашивающий данные, другую — целевой объект (поставщик данных). Впрочем, чтение и запись — не единственные операции, которые выполняются на уровне транзакций. Этот уровень, в частности, расширяет возможности передачи пакетов, предоставляемые канальным уровнем. Каждая полоса на уровне каналов подразделяется на несколько (числом до восьми) **виртуальных каналов**, по каждому из которых передаются данные того или иного типа. На уровне транзакций пакеты маркируются согласно классу трафика, определяющему ряд свойств, таких как «высокий приоритет», «низкий приоритет», «запрет слежения», «допускается доставка вне последовательности» и т. д. Выстраивая порядок обработки пакетов, коммутатор, помимо прочего, основывается на информации из маркеров.

Любая транзакция проходит в одном из четырех адресных пространств:

- пространство памяти (при выполнении стандартных операций чтения и записи);
- пространство ввода-вывода (для адресации регистров устройств);
- конфигурационное пространство (для инициализации системы и т. д.);
- пространство сообщений (для отправки сигналов, прерываний и т. д.).

Пространства памяти и ввода-вывода аналогичны традиционным — тем, что реализованы в современных системах. В конфигурационном пространстве возможна реализация разного рода механизмов, например автоматического конфигурирования (PnP). Пространство сообщений принимает на себя функции многочисленных ныне управляющих сигналов. Обойтись без этого пространства нельзя, ведь в PCI Express отсутствуют предусмотренные в шине PCI линии управления.

Программный уровень выступает посредником между PCI Express и операционной системой. Помимо прочего, на нем предусмотрен режим эмуляции шины PCI, позволяющий устанавливать в компьютерах, оснащенных PCI Express, старые операционные системы без каких-либо изменений. Естественно, при работе в таких условиях реализация всех возможностей PCI Express неосуществима, однако обеспечение обратной совместимости является необходимой мерой — по крайней мере, до того момента, пока во всех операционных системах не будет полностью реализована поддержка PCI Express. Опыт показывает, что этот процесс займет немало времени.

Информационный поток, характерный для PCI Express, показан на рис. 3.53, *б*. Команда, поступающая на программный уровень, передается на уровень транзакций, где из нее формируются заголовок и полезная нагрузка. Затем эти компоненты отправляются на канальный уровень, на котором в заголовке пакета устанавливается порядковый номер, а в хвостовике — CRC-код. Далее расширенный пакет передается на физический уровень, где с обоих концов к нему добавляются параметры кадра, и получившийся в результате физический пакет передается от отправителя получателю. На стороне получателя происходит обратный процесс — заголовок и хвостовик кадра канального уровня удаляются, а результат передается на уровень транзакций.

Концепция присоединения дополнительных данных на каждом последующем уровне стека протоколов применяется в компьютерных сетях уже очень долго и успешно. Основное отличие между сетевыми технологиями и PCI Express заключается в том, что в первом случае код, действующий на различных уровнях стека, почти всегда является программным и управляется операционной системой. В PCI Express, напротив, операции на всех уровнях реализуются аппаратно.

Структура PCI Express довольно сложна. Ее подробное описание имеется в работах [Mayhewand Krishnan, 2003; Solari and Congdon, 2005]. К тому же технология продолжает развиваться — в 2007 году была выпущена версия PCIe 2.0. Она поддерживает скорость передачи 500 Мбайт/с на канал и до 32 каналов, так что общая пропускная способность достигает 16 Гбайт/с. В 2011 году вышла версия PCIe 3.0, в которой кодировка изменилась с 8/10-разрядной на 128/130-разрядную, с возможностью выполнения до 8 миллиардов транзакций в секунду — вдвое больше, чем у PCIe 2.0.

Шина USB

Шины PCI и PCI Express очень хорошо подходят для соединения высокоскоростных периферийных устройств, но использовать интерфейс PCI для низкоскоростных устройств ввода-вывода (например, мыши и клавиатуры) было бы неэффективно. Изначально каждое стандартное устройство ввода-вывода соединялось с компьютером особым образом, при этом для добавления новых устройств использовались свободные ISA- и PCI-слоты. К сожалению, такая схема имеет некоторые недостатки.

Например, каждое новое устройство ввода-вывода часто оснащается собственной платой ISA или PCI. Пользователь при этом должен сам установить переключатели и перемычки на плате и удостовериться, что настроенная плата не конфликтует с другими платами. Затем пользователь должен открыть системный блок, аккуратно вставить плату, закрыть системный блок и включить компьютер. Для многих этот процесс очень сложен и часто приводит к ошибкам. Кроме того, число ISA- и PCI-слотов очень мало (обычно два или три). Автоматически конфигурируемые (PnP) платы исключают необходимость установки переключателей, но пользователь все равно должен открывать компьютер и вставлять туда плату. К тому же количество слотов шины ограничено.

В 1993 году представители семи компаний (Compaq, DEC, IBM, Intel, Microsoft, NEC и Nothern Telecom) собрались вместе, чтобы разработать шину, оптимально подходящую для подсоединения низкоскоростных устройств. Потом к ним примкнули сотни других компаний. Результатом их работы стала шина **USB** (Universal Serial Bus — **универсальная последовательная шина)**, которая сейчас широко используется в персональных компьютерах. Более подробное описание USB приведено в литературе [Anderson, 1997; Tan 1997].

Некоторые требования, изначально составившие основу проекта:

- пользователи не должны устанавливать переключатели и перемычки на платах и устройствах;
- пользователи не должны открывать компьютер, чтобы установить новые устройства ввода-вывода;
- должен существовать только один тип кабеля, подходящий для соединения всех устройств;
- устройства ввода-вывода должны получать питание через кабель;
- должна быть возможность подсоединения к одному компьютеру до 127 устройств;
- система должна поддерживать устройства реального времени (например, звуковые устройства, телефон);
- должна быть возможность устанавливать устройства во время работы компьютера;
- установка нового устройства не должна требовать перезагрузки компьютера;
- производство новой шины и устройств ввода-вывода для нее не должны требовать больших затрат.

Шина USB удовлетворяет всем этим условиям. Она разработана для низкоскоростных устройств (клавиатур, мышей, фотоаппаратов, сканеров, цифро-

вых телефонов и т. д.). Общая пропускная способность первой версии шины (USB 1.0) составляла 1,5 Мбит/с. Версия 1.1 работает на скорости 12 Мбит/с, что вполне достаточно для принтеров, цифровых камер и многих других устройств. Версия 2.0 поддерживает устройства со скоростью до 480 Мбит/с, достаточной для поддержки внешних дисков, веб-камер высокого разрешения и сетевых интерфейсов. В недавно появившейся версии USB 3.0 скорость возросла до 5 Гбит/с; только время покажет, какие новые и требовательные приложения породит этот сверхскоростной интерфейс.

Шина USB состоит из **корневого хаба** (root hub), который вставляется в разъем главной шины (см. рис. 3.49). Этот корневой хаб (часто называемый корневым концентратором) содержит разъемы для кабелей, которые могут подсоединяться к устройствам ввода-вывода или к дополнительным хабам, чтобы увеличить количество разъемов. Таким образом, топология шины USB представляет собой дерево с корнем в корневом хабе, который находится внутри компьютера. Коннекторы кабеля со стороны устройства отличаются от коннекторов со стороны хаба, чтобы пользователь случайно не подсоединил кабель другой стороной.

Кабель состоит из четырех проводов: два из них предназначены для передачи данных, один — для питания (+5 В) и один — для земли. Система передает 0 изменением напряжения, а 1 — отсутствием изменения напряжения, поэтому длинная последовательность нулевых битов порождает поток регулярных импульсов.

При подключении нового устройства ввода-вывода корневой хаб обнаруживает этот факт и прерывает работу операционной системы. Затем операционная система опрашивает новое устройство, выясняя, что оно собой представляет и какая пропускная способность шины для него требуется. Если операционная система решает, что для этого устройства пропускной способности достаточно, она приписывает ему уникальный адрес (1–127) и загружает этот адрес и другую информацию в конфигурационные регистры внутри устройства. Таким образом, новые устройства могут подсоединяться «на лету», при этом пользователю не нужно устанавливать новые платы ISA или PCI. Неинициализированные платы начинаются с адреса 0, поэтому к ним можно обращаться. Многие устройства снабжены встроенными сетевыми концентраторами для дополнительных устройств. Например, монитор может содержать два хаба для правой и левой колонок.

Шина USB представляет собой ряд каналов между корневым хабом и устройствами ввода-вывода. Каждое устройство может разбить свой канал максимум на 16 подканалов для различных типов данных (например, аудио и видео). В каждом канале или подканале данные перемещаются от корневого хаба к устройству и обратно. Между двумя устройствами ввода-вывода обмена информацией не происходит.

Ровно через каждую миллисекунду (±0,05 мс) корневой хаб передает новый кадр, чтобы синхронизировать все устройства во времени. Кадр состоит из пакетов, первый из которых передается от хаба к устройству. Следующие пакеты кадра могут передаваться в том же направлении, а могут и в противоположном (от устройства к хабу). На рис. 3.54 показаны четыре последовательных кадра.

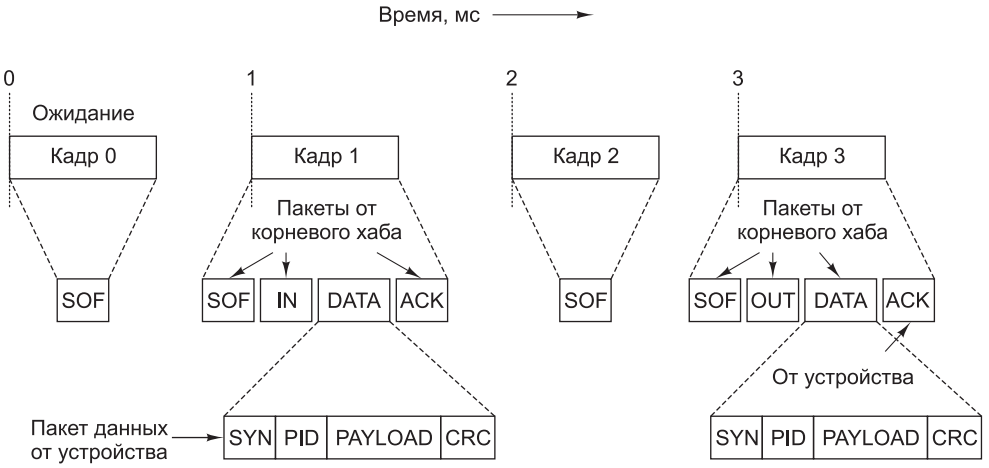


Рис. 3.54. Корневой хаб шины USB передает кадры каждую миллисекунду

В кадрах 0 и 2 не происходит никаких действий, поэтому в них содержится только пакет SOF (Start of Frame — начало кадра). Этот пакет всегда посылается всем устройствам. Кадр 1 — упорядоченный запрос (например, сканеру посылается запрос на передачу битов сканированного им изображения). Кадр 3 состоит из данных, передаваемых какому-нибудь устройству (например, принтеру).

Шина USB поддерживает 4 типа кадров: кадры управления, изохронные кадры, кадры передачи больших массивов данных и кадры прерывания. Кадры управления используются для конфигурирования устройств, передачи устройствам команд и запросов об их состоянии. Изохронные кадры предназначены для устройств реального времени (микрофонов, акустических систем и телефонов), которые должны принимать и посылать данные через равные временны́е интервалы. Задержки хорошо прогнозируются, но в случае ошибки такие устройства не производят повторной передачи. Кадры следующего типа используются для передач большого объема от устройств и к устройствам (например, принтерам) без требований, характерных для устройств реального времени. Наконец, кадры последнего типа нужны потому, что шина USB не поддерживает прерываний. Например, чтобы не вызывать прерывание всякий раз, когда нажимается клавиша, операционная система может опрашивать ее каждые 50 миллисекунд и «собирать» все необработанные данные о нажатии клавиш.

Кадр состоит из одного или нескольких пакетов. Пакеты могут посылаться в обоих направлениях. Существуют четыре типа пакетов: маркеры, пакеты данных, пакеты квитирования и специальные пакеты. Маркеры передаются от концентратора к устройству и предназначены для управления системой. Пакеты SOF, IN и OUT на рис. 3.54 — маркеры. Пакет SOF (Start of Frame — начало кадра) является первым в любом кадре, он идентифицирует начало кадра. Если никаких действий выполнять не нужно, пакет SOF — единственный в кадре. Пакет IN — это запрос. Этот пакет требует, чтобы устройство выдало определенные данные. Поля в пакете IN содержат информацию о том, какой именно канал запрашивается, и по этой информации устройство определяет, какие именно данные выдавать (если оно манипулирует несколькими потоками данных).

Пакет OUT объявляет, что далее последует передача данных для устройства. Последний тип маркера, SETUP (он не показан на рисунке), используется при конфигурировании.

Помимо маркеров существуют еще три типа пакетов. Это пакеты данных (используются для передачи 64 байт информации в обоих направлениях), пакеты квитирования и специальные пакеты. Формат пакета данных (DATA) показан на рис. 3.54. Он состоит из 8-разрядного поля синхронизации, 8-разрядного идентификатора типа пакета (PID), полезной нагрузки и 16-разрядного **CRC-кода** для обнаружения ошибок. Есть три типа пакетов квитирования: ACK (предыдущий пакет данных принят правильно), NAC (найдена ошибка CRC-кода) и STALL (устройство занято, ждите,).

А теперь давайте снова посмотрим на рис. 3.54. Корневой хаб должен отсылать по кадру каждую миллисекунду, даже если не происходит никаких действий. Кадры 0 и 2 содержат только один пакет SOF, который говорит о том, что ничего не происходит. Кадр 1 реализует опрос, поэтому начинается с пакетов SOF и IN, которые передаются от компьютера к устройству ввода-вывода, затем следует пакет DATA от устройства к компьютеру. Пакет ACK сообщает устройству, что данные были получены без ошибок. В случае ошибки устройство получает пакет NACK, после чего данные передаются заново (отметим, что изохронные данные повторно не передаются). Кадр 3 похож по структуре на кадр 1, но в нем поток данных направлен от компьютера к устройству.

После того как в 1998 году стандарт USB был окончательно утвержден, разработчики приступили к созданию следующей, высокоскоростной версии USB, названной **USB 2.0**. Этот стандарт во многом аналогичен USB 1.1 и совместим с ним, однако к двум прежним скоростям в нем добавляется новая — 480 Мбит/с. Все прочие изменения, включая реализацию нового интерфейса между корневым хабом и контроллером, не так существенны. В стандарте USB 1.1 было предусмотрено два интерфейса UHCI и OHCI. Интерфейс **UHCI** (Universal Host Controller Interface — **универсальный интерфейс хост-контроллера**) разработала компания Intel, переложив большую часть забот на программистов (читай — на Microsoft). Программисты вернули должок и выпустили интерфейс **OHCI** (Open Host Controller Interface — **открытый интерфейс хост-контроллера**), взвалив основную работу на разработчиков аппаратуры (читай — Intel). В процессе разработки стандарта USB 2.0 стороны пришли к взаимоприемлемому решению, выпустив новый интерфейс под названием **EHCI** (Enhanced Host Controller Interface — **усовершенствованный интерфейс хост-контроллера**).

Поскольку шина USB теперь передает данные со скоростью 480 Мбит/с, она становится серьезным конкурентом последовательной шины IEEE 1394 (FireWire), работающей на скорости 400 Мбит/с или 800 Мбит/с. Так как почти все современные системы на базе Intel оснащены шиной USB 2.0 или USB 3.0 (см. ниже), стандарт 1394 скоро уйдет в прошлое, причем основной причиной его вымирания будет не устаревание, а борьба за сферы влияния. USB — продукт компьютерной отрасли, а стандарт 1394 появился в мире потребительской электроники. Когда дело доходит до подключения камеры к компьютеру, каждая сторона желает использовать свой интерфейс. Похоже, «компьютерщики» в этой борьбе победили.

Через восемь лет после выхода USB 2.0 был анонсирован стандарт интерфейса USB 3.0. USB 3.0 поддерживает потрясающую пропускную способность

5 Гбит/с, хотя максимальная скорость, вероятно, будет достигаться только при использовании кабелей профессионального уровня. Устройства USB 3.0 структурно идентичны более ранним устройствам USB, и они полностью реализуют стандарт USB 2.0. Таким образом, при подключении к разъему USB 2.0 устройство USB 3.0 будет работать нормально.

Интерфейсы

Обычная компьютерная система малого или среднего размера состоит из микросхемы процессора, микросхем памяти и нескольких устройств ввода-вывода. Все эти микросхемы соединены шиной. Иногда все эти устройства интегрируются в однокристальную систему, как в случае TI OMAP4430. Мы уже рассмотрели память, центральные процессоры и шины. Теперь настало время изучить интерфейсы ввода-вывода. Именно через эти микросхемы компьютер обменивается информацией с внешними устройствами.

Интерфейсы ввода-вывода

В настоящее время существуют множество различных интерфейсов ввода-вывода, причем постоянно появляются новые интерфейсы. Из наиболее распространенных можно назвать UART, USART, контроллеры CRT, дисковые контроллеры и PIO. **UART (**Universal Asynchronous Receiver Transmitter — **универсальный асинхронный приемопередатчик)** — интерфейс ввода-вывода, который может считать байт из шины данных и побитно передать этот байт в линию последовательной передачи к терминалу или от терминала. Скорость работы микросхем UART различна: от 50 до 19 200 бит/с; ширина символа от 5 до 8 бит; 1, 1,5 или 2 стоповых бита; с проверкой на четность или на нечетность, или без нее — все управляется программно. **USART (**Universal Synchronous Asynchronous Receiver Transmitter — **универсальный синхронно-асинхронный приемопередатчик)** может осуществлять синхронную передачу, используя ряд протоколов, а также поддерживает все функции микросхемы UART. Так как с отмиранием телефонных модемов интерфейс UART уже не играет заметной роли, в качестве примера микросхемы ввода-вывода мы рассмотрим параллельный интерфейс PIO.

Интерфейсы PIO

Типичным примером интерфейса **PIO (**Parallel Input/Output — **параллельный ввод-вывод)** является микросхема Intel 8255A (рис. 3.55). Она содержит 24 линии ввода-вывода и может сопрягаться с любыми устройствами цифровой логики (например, клавиатурами, коммутаторами, индикаторами, принтерами). Программное обеспечение центрального процессора может записать 0 или 1 на любую линию или считать входное состояние любой линии, обеспечивая высокую гибкость. Небольшая система на базе процессора, использующая интерфейс PIO, может управлять разнообразными физическими устройствами — роботами, тостерами, электронными микроскопами. Чаще всего интерфейсы PIO встречаются во встроенных системах.

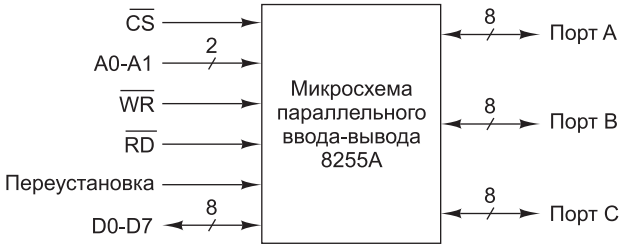


Рис. 3.55. Микросхема 8255A

Конфигурация интерфейса PIO определяется 3-разрядным регистром, который указывает, должны ли три независимых 8-разрядных порта использоваться для ввода (0) или вывода цифрового сигнала (1). Запись соответствующего значения в регистр конфигурации позволяет осуществлять произвольную комбинацию ввода-вывода по трем портам. С каждым портом связан 8-разрядный регистр. Чтобы настроить линии на порт, центральный процессор записывает 8-разрядное число в соответствующий регистр, и это 8-разрядное число появляется на выходных линиях и остается там до тех пор, пока регистр не будет перезаписан. Чтобы использовать порт для ввода, центральный процессор просто считывает соответствующий регистр.

Возможно построение и более сложных интерфейсов PIO. Например, один популярный режим предусматривает квитирование связи с внешними устройствами. Например, чтобы передать данные устройству, микросхема 8255A может передать данные в порт вывода и подождать, пока устройство не выдаст сигнал о том, что данные получены и можно посылать еще. В микросхему включены необходимые логические схемы для фиксации таких сигналов и передачи их центральному процессору.

Из функциональной диаграммы PIO на рис. 3.55 мы видим, что помимо 24 выводов для трех портов микросхема 8255A содержит восемь линий, непосредственно связанных с шиной данных, линию выбора элемента памяти, линии чтения и записи, две адресные линии и линию для переустановки микросхемы. Две адресные линии позволяют выбрать один из четырех внутренних регистров, три из которых соответствуют портам A, B и C, а четвертый регистр является регистром конфигурации. Обычно две адресные линии соединяются с двумя младшими битами адресной шины. Линия выборки позволяет строить из 24-разрядных интерфейсов PIO интерфейсы большей разрядности, с добавлением дополнительных адресных линий и использования их для выбора нужного интерфейса посредством установки линии выборки микросхемы.

Декодирование адреса

До настоящего момента мы не останавливались подробно на том, как происходит выбор микросхемы памяти или устройства ввода-вывода. Рассмотрим простой 16-разрядный встроенный компьютер, состоящий из центрального процессора, памяти EPROM объемом 2 К × 8 байт для хранения программы, ОЗУ объемом 2 К × 8 байт для хранения данных и интерфейса PIO. Такая небольшая система может встраиваться в дешевую игрушку или простой прибор. После того как продукт пойдет в массовое производство, EEPROM заменяется обычным ПЗУ.

Выборка интерфейса PIO может осуществляться одним из двух способов: как устройства ввода-вывода или как части памяти. Если мы собираемся использовать PIO в качестве устройства ввода-вывода, то мы должны обратиться к нему по соответствующей линии шины, которая означает, что обращение относится к устройству ввода-вывода, а не к памяти. Другой подход называется **вводом-выводом с отображением на память** (memory-mapped I/O). В этом случае микросхеме требуется назначить 4 байта памяти для трех портов и регистра управления. Наш выбор в какой-то степени произволен. Рассмотрим ввод-вывод с отображением на память, поскольку этот подход наглядно иллюстрирует некоторые интересные проблемы сопряжения.

Памяти EPROM требуется 2 Кбайт адресного пространства, ОЗУ — также 2 Кбайт, PIO — 4 байта. Поскольку в нашем примере адресное пространство составляет 64K адресов, мы должны выбрать, где поместить указанные три устройства. Один из возможных вариантов показан на рис. 3.56. EPROM занимает адреса до 2K, ОЗУ — от 32K до 34K, PIO — 4 старших байта адресного пространства, от адресов 65 532 до 65 535. С точки зрения программиста не важно, какие именно адреса использовать, однако для сопряжения это имеет большое значение. Если бы мы обращались к PIO через пространство ввода-вывода, нам не потребовались бы адреса памяти (зато понадобились бы четыре адреса пространства ввода-вывода).

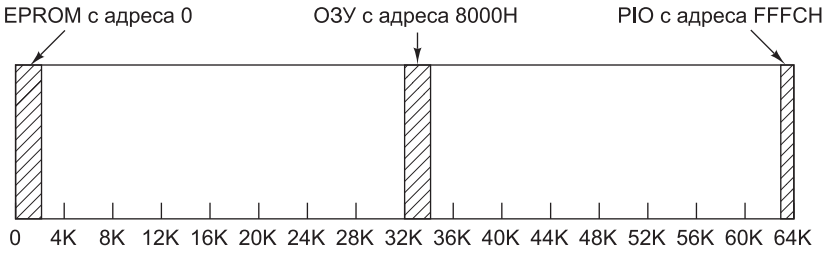


Рис. 3.56. Расположение EPROM, ОЗУ и PIO в пространстве из 64K адресов

При таком распределении адресов к EPROM нужно обращаться с помощью 16-разрядного адреса памяти 00000*xxxxxxxxxxx* (в двоичной системе). Другими словами, любой адрес, у которого пять старших битов равны 0, попадает в область памяти до 2K, то есть в EPROM. Таким образом, сигнал выбора EPROM можно связать с 5-разрядным компаратором, у которого один из входов всегда будет соединен с 00000.

Того же результата можно достичь более эффективно: с использованием вентиля ИЛИ с пятью входами, связанными с адресными линиями от A_{11} до A_{15}. Выходной сигнал может быть равен 0 тогда и только тогда, когда все пять линий равны 0. В этом случае устанавливается сигнал $\overline{CS}$. Этот метод адресации показан на рис. 3.57, *а*; он называется полным декодированием адреса.

Тот же принцип можно применить и для ОЗУ. Однако ОЗУ должно отзываться на бинарные адреса типа 10000*xxxxxxxxxxx*, поэтому необходим дополнительный инвертор (он показан на схеме). Декодирование адреса PIO несколько сложнее, поскольку он выбирается с помощью 4-х адресов типа 1111111111111*xx*. Один из возможных вариантов схемы, которая устанавливает сигнал $\overline{CS}$ только в том случае, если на адресной шине появляется адрес данного типа, показан на

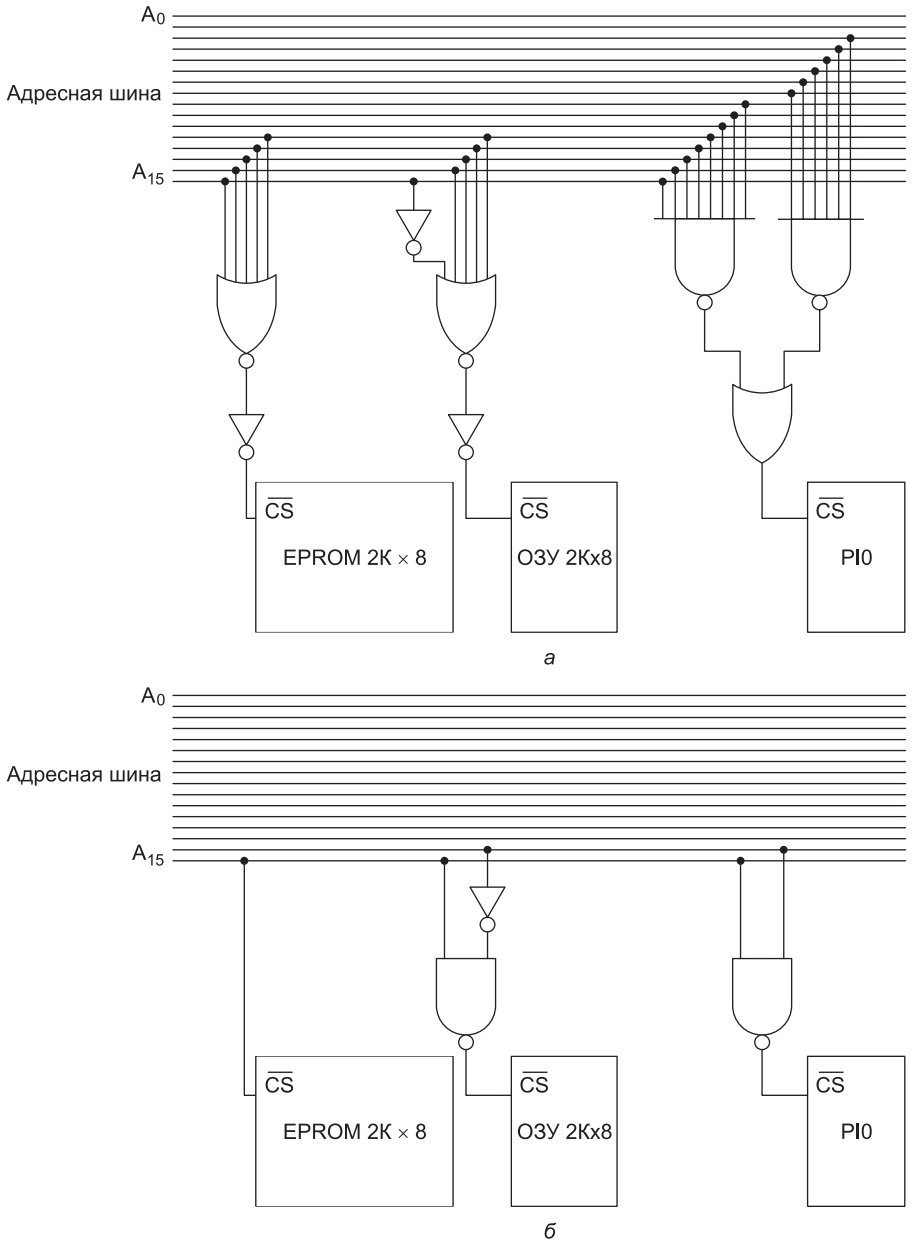


Рис. 3.57. Полное декодирование адреса (*а*); частичное декодирование адреса (*б*)

рисунке. Здесь используются два вентиля НЕ-И с восемью входами, которые соединяются с вентилем ИЛИ.

Если компьютер состоит только из центрального процессора, двух микросхем памяти и PIO, декодирование адреса значительно упрощается. Дело в том, что у всех адресов EPROM (и только у EPROM) старший бит A_{15} всегда равен 0.

Следовательно, мы можем просто связать сигнал $\overline{CS}$ с линией A_{15}, как показано на рис. 3.58, *б*.

Теперь решение разместить ОЗУ с адреса 8000H кажется не таким уж произвольным. Отметим, что в ОЗУ попадают адреса типа 10*xxxxxxxxxxxxxx*, поэтому для декодирования достаточно двух бит. Точно так же, любой адрес, начинающийся с 11, является адресом PIO. Теперь полная логика декодирования состоит из двух вентилей НЕ-И и инвертора.

Логика, которую иллюстрирует рис. 3.58, *б*, называется **частичным декодированием адреса**, поскольку в данном случае полные адреса не используются. При таком декодировании считывание из адресов 0001000000000000, 0001100000000000 и 0010000000000000 дает один и тот же результат. В действительности любой адрес в нижней половине адресного пространства означает выбор EPROM. Поскольку дополнительные адреса не используются, в этом нет ничего ужасного, но при разработке компьютера, который в будущем предполагается расширять (в случае с игрушками это маловероятно), следует избегать частичного декодирования, поскольку оно значительно ограничивает адресное пространство.

Можно применять и другую технологию декодирования адреса — с использованием декодера (см. рис. 3.12). Связав три входа с тремя адресными линиями самых старших разрядов, мы получаем восемь выходов, которые соответствуют адресам в первом отрезке 8К, втором отрезке 8К и т. д. В компьютере, содержащем 8 микросхем ОЗУ по 8К × 8 байт, полное декодирование осуществляет одна такая микросхема. Если компьютер содержит 8 микросхем памяти по 2 К × 8 байт, для декодирования также достаточно одного декодера при условии, что каждая микросхема памяти занимает отдельную область адресного пространства в 8К. (Вспомните наше замечание о том, что расположение микросхем памяти и устройств ввода-вывода внутри адресного пространства имеет значение.)