СЕРГЕЙ ПЕТРОВ

bhv

ШИНЫ PCI, PCI Express

АРХИТЕКТУРА, ДИЗАЙН, ПРИНЦИПЫ ФУНКЦИОНИРОВАНИЯ

- ПРОТОКОЛ ШИНЫ PCI
- ПРАВИЛА АДРЕСАЦИИ И АРБИТРАЖА
- ПОРЯДОК СЛЕДОВАНИЯ ТРАНЗАКЦИЙ
- ПАРАМЕТРЫ ИСПОЛЬЗУЕМЫХ СИГНАЛОВ
- ОРГАНИЗАЦИЯ КОНФИГУРАЦИОННОГО ПРОСТРАНСТВА
- ОСНОВНЫЕ ПРИНЦИПЫ ФУНКЦИОНИРОВАНИЯ ШИНЫ PCI EXPRESS

АППАРАТНЫЕ СРЕДСТВА

Сергей Петров

ШИНЫ PCI, PCI Express

АРХИТЕКТУРА, ДИЗАЙН, ПРИНЦИПЫ ФУНКЦИОНИРОВАНИЯ

Санкт-Петербург
«БХВ-Петербург»
2006

УДК 681.3.06
ББК 32.973.26
П30

Петров С. В.

П30 Шины PCI, PCI Express. Архитектура, дизайн, принципы функционирования. — СПб.: БХВ-Петербург, 2006. — 416 с.: ил.

ISBN 5-94157-383-9

В книге обобщены материалы комплекта спецификаций шин PCI и PCI Express. Рассмотрены все аспекты разработки устройств, приведена информация о конфигурационном пространстве и правилах работы с ним. Описаны требования, предъявляемые к электронным компонентам. Приведен протокол шины PCI: адресация, правила передачи, порядок следования и завершения транзакций, арбитраж и т. д. Объяснен механизм задержанных транзакций. Рассмотрены вопросы обеспечения помехоустойчивости и надежности, возможности по расширению до 64 разрядов. Описана организация регистров и структура записей 256-байтного пространства конфигурации. Раскрыты основные понятия и определения шины PCI Express, ее архитектура, принципы функционирования и обратная совместимость с шиной PCI.

Для разработчиков аппаратуры, программистов и студентов профильных специальностей

УДК 681.3.06
ББК 32.973.26

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Юрий Рожко*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

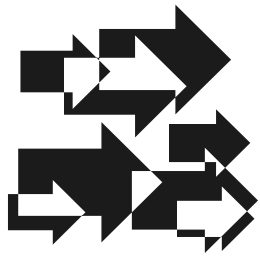
Лицензия ИД № 02429 от 24.07.00. Подписано в печать 26.05.06.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 33,54.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-383-9

Оглавление



ГЛАВА 1

Введение в шину PCI

История PCI

Одним из основных требований, предъявляемых к вычислительным системам, является обеспечение необходимого быстродействия. При использовании стандартной последовательной архитектуры общая производительность определяется самым "медленным" компонентом системы. Пропускная способность канала связи является фактором, более всего влияющим на быстродействие, т. к. от него зависит скорость обмена данными. В случае взаимодействия процессора и какого-либо внешнего устройства в качестве такого канала связи выступает *шина расширения*, иначе называемая *локальной*.

Шина расширения используется для объединения отдельных устройств или их групп. При разработке шины необходимо учитывать их физические параметры, назначение и функциональные особенности. Такие характеристики системы, как быстродействие, функциональная однородность, общность электрических характеристик, зависят от локальной шины. Она также определяет, какие устройства являются совместимыми.

Постепенно шина расширения превратилась из набора проводников в самостоятельный интерфейс. По мере развития локальных шин на практике были проверены различные идеи, инженерные решения, большинство из которых использовались при разработке стандарта PCI.

Развитие шинной архитектуры

Одной из первых попыток введения стандартов в области шин расширения был стандарт *PC*, предложенный фирмой IBM. Хотя, по сравнению с другими шинами расширения, использовавшимися в то время, шина PC не предложи-

ла ничего оригинального, все необходимое она реализовывала, и вскоре заняла лидирующее положение.

Примечание

Например, стандарт S-100 позволял передавать информацию по 16 линиям, в то время как шина PC имела только 8 линий данных. Иначе эта шина называлась *ISA-8* (Industry Standard Architecture).

Обычно все проводники в шине имеют один и тот же уровень сигнала на всем своем протяжении, что и было реализовано в шине PC. Восьмой проводник, ближайший к контактам питания в IBM PC XT, отличается по своим электрическим характеристикам от всех других проводников шины. Этот проводник был помечен как резервный в оригинальной шине PC и в последующем предназначался для реализации функции выбора платы.

За три года, прошедших между появлением IBM PC и IBM PC AT, недостатки конструкции шины PC стали явными. Не удовлетворяли ограничения, накладываемые шиной на работу с памятью, не отвечало потребностям и количество линий данных. Кроме того, многие функции, реализуемые системой (а не шиной), переросли простое, однозначное применение.

В компьютере IBM PC AT шина PC получила дальнейшее развитие. Она отличалась наличием дополнительного разъема, который позволял передавать больше информации и расширял адресные линии. Шина получила название *AT* или, иначе, *ISA-16*. Число адресных линий было увеличено на 4 и линий данных на 8, т. е. шина обладала 16 линиями данных и 24 адресными линиями, что позволяло ей работать с 16 Мбайт памяти — пределом адресации микропроцессора 80286.

При переходе от 16-битной шины расширения к 32-битной компания IBM предложила новую шину, явившуюся частью первых компьютеров Personal System/2 (PS/2). Она настолько отличалась от шин PC и AT, что IBM дала всей системе новое имя — микроканальная архитектура. Шина *MCA* (Micro Channel Architecture) или, иначе, *"микроканал"* (Microchannel) является стандартом 32-битной шины расширения, принятым только IBM [22].

Примечание

Нечто подобное сделала Intel при разработке своих первых плат для 80386 систем. Системные платы Intel характеризовались комбинацией 8-, 16-, 32-битных разъемов расширения. Тем не менее Intel отказалась от разработки собственной 32-битной шины.

Группа компаний, возглавляемая Compaq Computer Corporation, разработала 32-битный стандарт шины независимо от микроканала. Его презентация состоялась 13 сентября 1988 года. Стандарту дали имя расширенной стандартной архитектуры (*EISA* — Extended Industry Standart Architecture). EISA мож-

но рассматривать как доработку шины AT (ISA). Его характеристики близки к микроканалу, но сильно отличаются по способу реализации.

Последующее развитие элементной базы и микропроцессорных архитектур привело к тому, что шины ISA, EISA перестали отвечать требованиям по пропускной способности. При взаимодействии устройства с процессором запросу на обслуживание нужно было пройти через расширительный мост, шину памяти, кэш и локальную шину процессора (рис. 1.1).

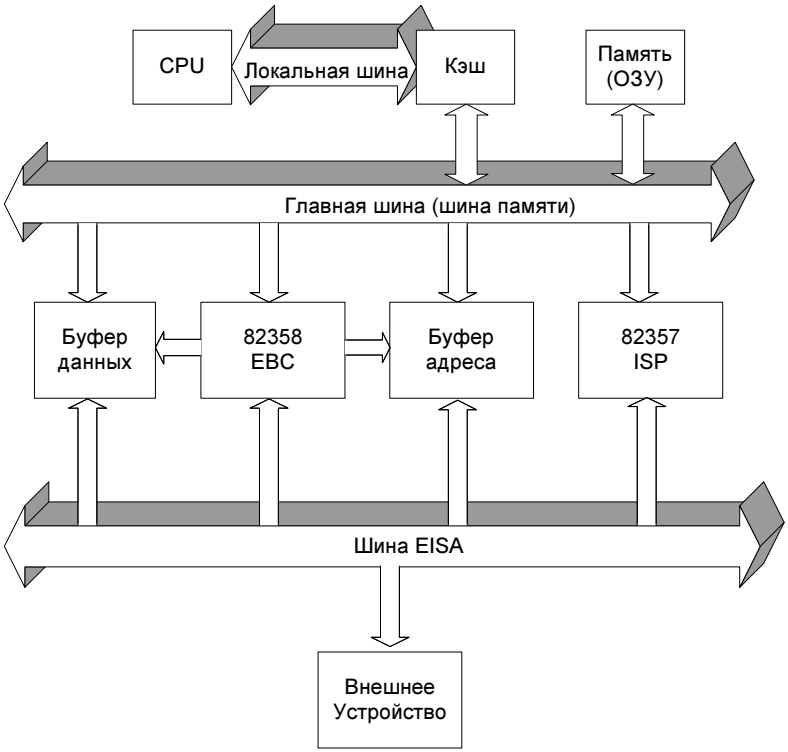


Рис. 1.1. Пример EISA архитектуры

Все это приводило к значительным задержкам при обработке запроса и операциях ввода-вывода. Требовалась новая идея, новый подход, реализующий следующее условие: локальная шина, обеспечивающая возможность обмена данными с периферийными устройствами, должна обладать пропускной способностью, близкой к пропускной способности системного процессора.

Разработка шины *PCI* (Peripheral Component Interconnect, подсоединение периферийных компонентов) началась весной 1991 года как внутренний проект корпорации Intel (шина спецификации PCI 0.1). Специалисты компании поставили перед собой цель разработать недорогое решение, которое позволило

бы полностью реализовать возможности нового поколения процессоров 486/Pentium/P6. Особенно подчеркивалось, что разработка проводилась "с нуля", а не была попыткой установки новых "заплат" на существующие решения. В результате шина PCI появилась в июне 1992 года (спецификация PCI 1.0). Разработчики Intel отказались от использования шины процессора и ввели еще одну, локальную шину. Массовое применение PCI началось в Pentium-системах, но она использовалась и с 486-ми процессорами. Частота шины составляла от 20 до 33 МГц, а теоретическая максимальная скорость передачи данных составляет 132 Мбит/с для 32-разрядной шины и 264 Мбит/с для 64-разрядной [17].

Стандарт PCI был объявлен открытым и передан комитету PCI Special Interest Group, который продолжил работу по совершенствованию шины. Выгоды установления открытого стандарта для шин системного ввода-вывода хорошо видны на примере производства IBM PC. Важно отметить, что стандарт PCI для локальных шин установлен для упрощения проектирования, снижения стоимости и расширения ассортимента отдельных компонентов локальной шины, а также плат расширения [17]. Преимущества, обеспечиваемые отдельными разработками локальной шины, послужили причиной появления нескольких версий ее реализации.

Особенности PCI версии 2.3

По сравнению с предыдущими версиями спецификаций (версий 1.0, 2.0, 2.1, 2.2), в PCI версии 2.3 можно выделить следующие особенности:

- SMBus — добавлена поддержка для двухпроводного интерфейса управления SMBus (System Management Bus — шина системного управления);
- время сброса — установлены дополнительные требования к временным параметрам для сброса состояния сигнала;
- размер платы расширения — одним из классификационных признаков плат расширения является их размер, иначе *форм-фактор*. Ранее были определены 3 физических типа плат: длинный, короткий и инвариантный. В версии 2.3 добавлен новый тип — LOW-PROFILE;
- добавлены новые коды классов, новые идентификаторы типа устройств;
- не поддерживаются платы расширения 5,0 В с аппаратным ключом;
- добавлено поле "Interrupt Disable" в регистр команд и "Interrupt Status" в регистр состояния;
- совместимость плат расширения — спецификация PCI и приложение к ней PCI-X определяют платы, которые функционируют на различных частотах и с различными протоколами. Все платы расширения 3,3 В и компоненты,

включая PCI 33 МГц, 66 МГц, PCI-X 66 МГц и PCI-X 133 МГц спроектированы для работы в системе PCI 33 МГц.

При разработке системных плат не нужно специально обеспечивать поддержку всех типов плат расширения. Совместимость сигнальных сред 3,3 и 5,0 В обеспечивается за счет наличия двух типов плат расширения, различающихся рабочими напряжениями. Определена "универсальная" плата, которая включает разъемы 3,3 и 5,0 В, и плата расширения "3.3 В", которая включает только разъем 3,3 В.

Методы коммутации

При определенных условиях электромагнитная волна обладает способностью отражаться. Такое явление может происходить, если цепь разомкнута или замкнута на конце накоротко. В этом случае электромагнитная волна, достигнув конца цепи, отражается полностью и распространяется к ее началу.

При сопротивлении нагрузки, которое отличается от волнового сопротивления цепи (несогласованное включение), энергия будет отражаться частично: чем меньше будет отличаться сопротивление нагрузки от волнового сопротивления цепи, тем меньше будет отражаться энергии и больше поглощаться нагрузкой [19].

При сопротивлении нагрузки, равном волновому сопротивлению цепи (согласованное включение), электромагнитная волна отражаться не будет, и вся энергия поступит в нагрузку.

Таким образом, при согласованном включении в цепи будут иметь место только электромагнитные волны, распространяющиеся от начала к концу цепи. Эти волны называются *прямыми (падающими)*. При несогласованном включении, кроме падающих волн, появляются *обратные (отраженные)* волны, распространяющиеся от конца к началу цепи [19].

При передаче тактового импульса его восходящий фронт, распространяясь по проводникам шины, нарастает выше порога переключения приемопередатчиков, подключенных к соответствующим проводникам шины. Несогласованное включение шинного драйвера с волновым сопротивлением цепи вызывает возникновение отраженной волны. Взаимодействие прямого и отраженного сигналов приводит к появлению некоторого уровня шума на восходящем фронте импульса и изменениям уровня сигнала в области порога переключения шинных приемников. В результате этого осуществление выбора линии приемником задерживается до тех пор, пока сигнал не стабилизируется. Шины с невысокой производительностью, например ISA, не учитывают влияния отраженных волн. Высокопроизводительные шины не могут допускать такой задержки. Для решения проблемы существует два метода, называемые *коммутацией на падающей волне* и *коммутацией на отраженной волне*.

Основная идея метода *коммутации на падающей волне* (Incident Wave Switching) состоит в следующем. Импульс необходимо считывать после его самого первого пересечения порога переключения приемника (т. е. на падающей волне), до возникновения колебаний. Такие линии содержат специальные *шинные драйверы*, обеспечивающие большой электрический ток при малом входном сопротивлении и цепи завершения, которые уменьшают мощность отраженной волны. Шина, разработанная на основе данного метода (рис. 1.2), обладает следующими характеристиками:

- связью на высокой скорости — полное быстродействие линии связи зависит от скорости распространения электромагнитных колебаний. Поскольку волна распространяется только в одном направлении, то достигается предельная скорость коммутации;
- качеством сигнала — чтобы избежать отражений, шина должна содержать цепь завершения, а в качестве шинного драйвера должен использоваться малый внутренний резистор. Такие параметры реализованы в шинных драйверах, которые способны обеспечить ток 48 мА или даже 64 мА. Кроме того, завершенные шины обладают хорошо определенным уровнем напряжения, что уменьшает ошибки, вызванные внешними наводками;

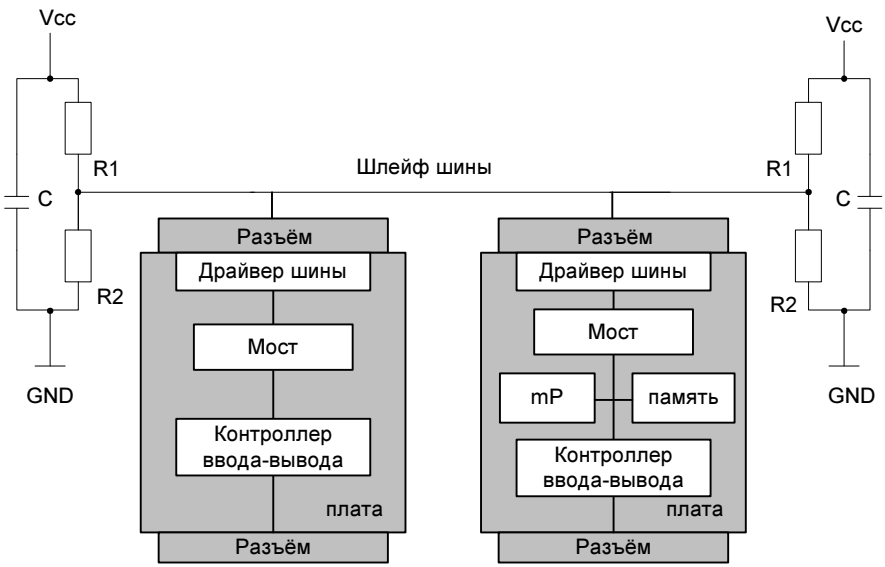


Рис. 1.2. Шина с коммутацией на падающей волне

- универсальностью — за счет устойчивого качества сигнала и высокой скорости на шине может быть интегрировано большое число разъемов. Например, шина VME (Versa Module Europa), предназначенная для объ-

единения устройств, работающих в режиме реального времени, допускает установку до 21 платы расширения;

- ❒ большим энергопотреблением — является главным недостатком данного метода, исключающим использование метода падающей волны для следующих применений:
  - мобильных систем — шинный драйвер представляет собой крупногабаритное устройство, как правило, с 8 или 12 линиями, что обусловлено высоким энергопотреблением. Его использование требует дополнительного места на плате, что увеличивает размер плат расширения. Таким образом, метод коммутации на падающей волне не предназначен для систем с питанием от аккумуляторных батарей или других мобильных применений;
  - проектов с низким потреблением и большим количеством линий — управление единственной линией 48 мА не создает трудностей. Однако параллельное управление 40—50 или более линиями увеличивает требования к электропитанию и охлаждению;
- ❒ высокой ценой — возрастание стоимости связано с расходами на шинный драйвер, завершающую цепь, дополнительное место на плате для всех этих компонентов и на охлаждение. Все это исключает использование метода падающей волны для низкобюджетных пользовательских проектов.

Метод *коммутации на отраженной волне* (Reflected-Wave Switching) использует обе, и падающую, и отраженную волны. При таком подходе на линии отсутствует завершающая цепь (рис. 1.3). Внутреннее сопротивление передатчика эквивалентно сопротивлению линии. В результате уровень сигнала падающей волны равен половине значения, необходимого для коммутации. Отраженная волна увеличивает напряжение до требуемого уровня (рис. 1.4).

Данный метод обладает следующими преимуществами:

- ❒ прямым соединением устройства и шины — специальный шинный драйвер не используется;
- ❒ низким энергопотреблением — возможность применения в мобильных системах с питанием от аккумуляторных батарей;
- ❒ снижением стоимости реализации шины — отсутствует завершающая цепь;
- ❒ компактным размером — уменьшаются размеры плат расширения.

Недостатки данного метода включают:

- ❒ ухудшение качества сигнала — цепи с низким напряжением питания более подвержены воздействию помех;

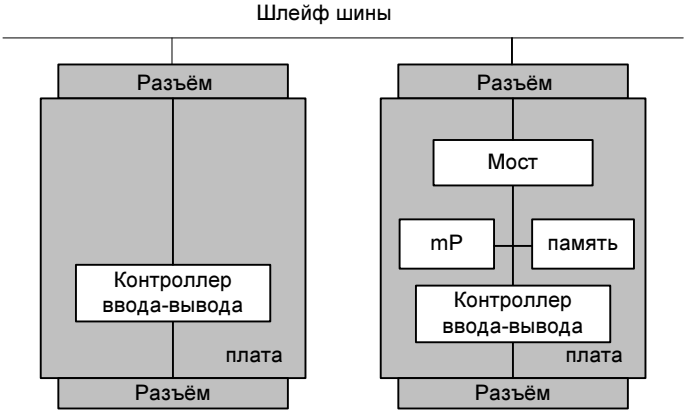


Рис. 1.3. Цепь коммутации на отраженной волне

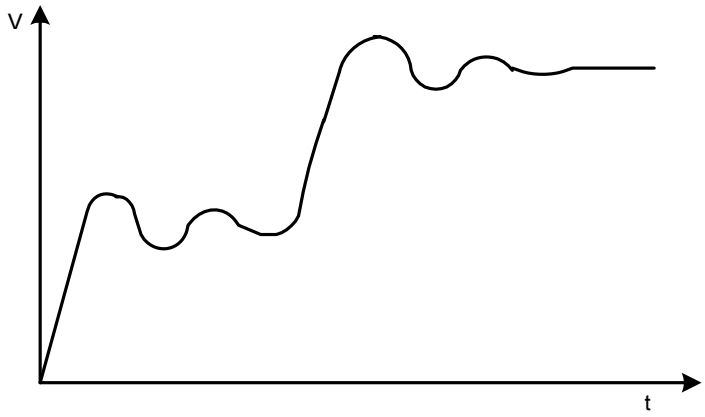


Рис. 1.4. Рост напряжения при отраженной волне

- снижение производительности — поскольку волна должна распространяться дважды, чтобы сигнал достиг необходимого уровня;
- уменьшение размерности — как следствие уменьшения длины шины; стандартная шина PCI 33 МГц допускает до 4-х слотов для плат расширения, увеличение частоты до 66 МГц приводит к ограничению до 2—3 слотов, а при работе шины на частоте 133 МГц допускается только один слот.

Шина PCI была первой, где использовался метод коммутации на отраженной волне. Как уже упоминалось, шина PCI разрабатывалась как недорогое решение с приемлемой производительностью.

Место шины PCI в вычислительной архитектуре

Шина PCI разрабатывалась как промышленный стандарт высокопроизводительной локальной шины. Такой стандарт должен был обеспечить минимальную стоимость реализации, предоставляя при этом определенную степень свободы производителям компонентов, системных плат и плат расширения. Основные усилия проектировщиков были направлены на достижение оптимального соотношения цена/качество, возможности применения для разных платформ и архитектур, а также учет будущих требований к вычислительным системам. На рис. 1.5 показаны применения локальной шины PCI.

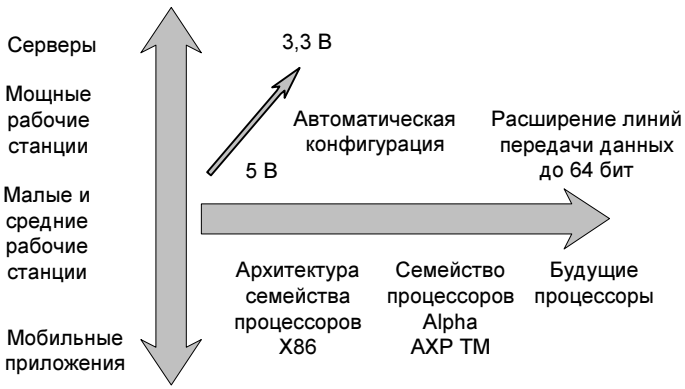


Рис. 1.5. Применения шины PCI

Первоначально шина PCI предназначалась для сектора высокопроизводительных настольных систем. Тем не менее шина применяется и в области мобильных приложений и серверов [9, 10, 11].

Назначение шины

Локальная шина PCI использует сигнальную среду 3,3 и 5,0 В. Однако со временем намечается постепенный переход к сигнальной среде 3,3 В (рис. 1.5). Так, например, в спецификации PCI, начиная с версии 2.3, платы расширения, имеющие питание 5,0 В с дополнительным аппаратным ключом, больше не поддерживаются.

Компоненты PCI и платы расширения являются процессорно-независимыми, что обеспечивает их совместимость с новыми поколениями процессоров, а также с многопроцессорными архитектурами.

Процессорная независимость позволяет оптимизировать шину для функций ввода-вывода, делает возможным конкурирующую работу локальной шины

с подсистемой процессор/память и допускает использование для графической подсистемы множества высокопроизводительных периферийных устройств (видеоадаптеров, SCSI-устройств, FDDI, жестких дисков и т. д.). Переход к новым видео- и мультимедийным дисплеям, а именно HDTV и трехмерным, продолжит повышение требований к разрядности локальной шины. Имеющееся "прозрачное" расширение 32-разрядных шин данных и адресов до 64 разрядов делает периферийные устройства совместимыми, в прямом и обратном направлении, с 32- и 64-разрядной локальной шиной PCI.

Также определена прямая и обратная совместимость спецификации PCI-X, увеличивая разрядность на частоте 33 МГц дополнительным форм-фактором.

Стандарт локальной шины обладает еще одним преимуществом. Для PCI-компонентов и плат расширения определены регистры конфигурации. Они обеспечивают автоматическую конфигурацию устройств на шине при включении питания [9, 10, 11].

Характеристики шины

Основные характеристики шины в порядке их значимости перечислены далее.

- ❒ Производительность.
  - Прозрачный переход от 32-разрядных данных к 64-разрядным данным с шиной, работающей на частотах 33, 66 и 133 МГц, с соответствующим увеличением скорости обмена данными.
  - Переменная длина блоков данных и переключаемый режим, как для чтения, так и для записи, повышающие производительность при работе с графикой.
  - Произвольный доступ с малым временем задержки (60 нс при записи для шины 33 МГц и 30 нс при записи из ведущего устройства в регистры ведомого для шины PCI-X).
  - Возможность полного распараллеливания подсистемы процессор/память.
  - Синхронизация шины 33, 66 или 133 МГц.
  - Скрытый (перекрываемый) центральный арбитраж.
- ❒ Цена.
  - Оптимизация для прямых "межкремниевых" соединений компонентов, т. е. отсутствие "склеивающей логики". Электрические и частотные спецификации реализованы в соответствии со стандартными технологиями разработки специализированных микросхем и другими типовыми процессами.

- Мультиплексированная архитектура снижает количество выводов (47 — для сигналов ведомого устройства, 49 — для ведущего) и уменьшает размеры корпуса PCI-компонентов, либо обеспечивает дополнительные функции в корпусе заданного размера.

❒ Использование.

- Поддержка полной автоконфигурации плат расширения и компонентов локальной шины. PCI-устройства имеют регистры, в которых хранится информация об устройстве, используемая для установки.

❒ Долговечность.

- Независимость от процессора. Поддерживается множество семейств процессоров, в том числе и будущие их поколения (через переходники-"мосты" или путем прямой интеграции).
- Поддержка 64-разрядной адресации.
- Поддержка использования как напряжения в 5,0 В, так и 3,3 В для сигналов. Специальные возможности делают переход между рабочими напряжениями 5,0 и 3,3 В более плавным.

❒ Взаимодействие/надежность.

- Малый форм-фактор плат расширения.
- Наличие сигналов, позволяющих оптимизировать напряжение питания для ожидаемой степени загруженности системы путем отслеживания работы плат расширения, что позволяет добиться от системы максимальной эффективности работы.
- Более 2000 часов электрического моделирования в программном пакете PSPICE с последующей коррекцией аппаратной модели.
- Прямая и обратная совместимость с 32- и 64-разрядными платами расширения и компонентами.
- Прямая и обратная совместимость с PCI 33 МГц, PCI 66 МГц, PCI-X 66 МГц и PCI-X 133 МГц платами расширения и компонентами.
- Повышенная надежность и расширенные возможности взаимодействия с платами расширения благодаря снижению требований к загрузке и частоте локальной шины на уровне компонентов, уничтожению буферов и "склеивающей логики".

❒ Гибкость.

- Возможности по полному управлению множеством PCI-мастеров, обеспечивающие одноранговый доступ любого PCI-мастера к любому другому PCI-мастеру либо к ведомому устройству.

- ❒ Целостность данных.
  - PCI обеспечивает контроль четности как для данных, так и для адресов, позволяя создавать устойчивые пользовательские платформы.
- ❒ Программная совместимость.
  - PCI-компоненты могут быть полностью совместимы с существующим программным обеспечением и драйверами устройств и могут применяться для различных классов платформ.

Пример PCI-системы

В соответствии со спецификацией PCI каждое устройство имеет доступ к локальной шине процессора и шине системной памяти через связывающий их мост (рис. 1.6).

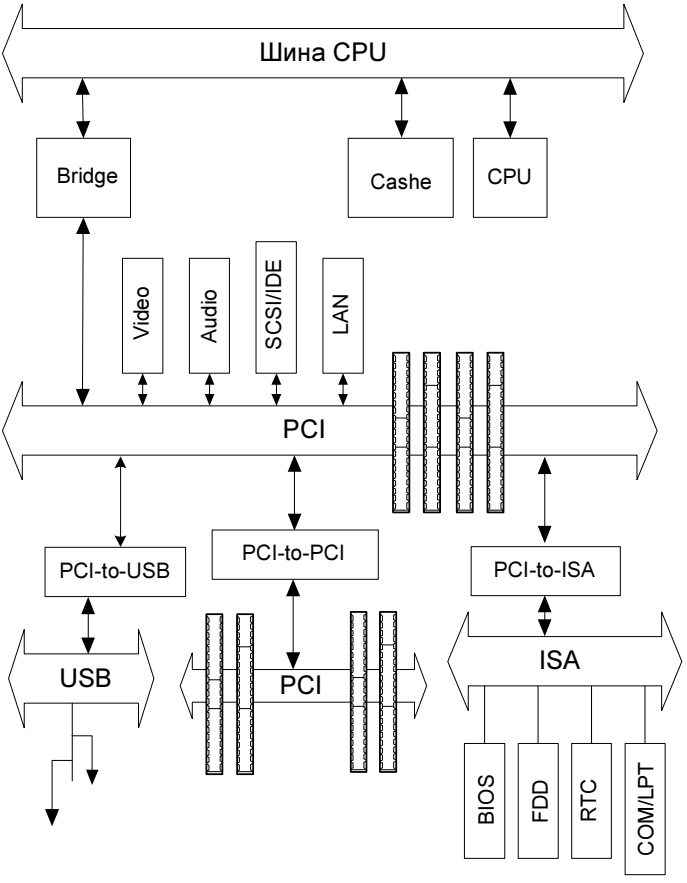


Рис. 1.6. Пример PCI-системы

Такое решение дает по крайней мере два преимущества. Во-первых, оно позволяет выполнять на шине несколько операций одновременно, например, процессор может забирать из кэш-памяти моста данные, в то время как устройство обращается к системной памяти. Во-вторых, обеспечивает независимость локальной шины от процессора [9, 10, 11].

Комплект стандартов и спецификаций

Полный комплект спецификаций PCI не включает такие стандарты, как PCI-X, CompactPCI и PCI Express. Тем не менее они так или иначе связаны как с основным документом — спецификацией шины, так и с дополнительными — PCI-to-PCI Bridge, Bus Power Management и т. д.

Спецификация шины PCI

Полный комплект стандартов включает набор спецификаций, описывающих различные аспекты разработки PCI. Основной документ-спецификация локальной шины PCI включает протокол, электрическую, механическую и конфигурационную спецификации для локальной шины PCI и плат расширения. Описания электрических сигналов приводятся для напряжений питания 3,3 и 5,0 В. Спецификация локальной шины PCI определяет аппаратное обеспечение PCI.

Более подробная информация, касающаяся системного проектирования PCI и спецификации PCI BIOS, предоставляется членам *PCI SIG* (PCI Special Interest Group, группа поддержки PCI). Это независимая ассоциация членов индустрии микроЭВМ, которая создана для контроля и дальнейшей разработки локальной шины PCI. В ее состав входят широко известные компании: IBM, Microsoft, AMD, Adaptec, Intel, Hewlett-Packard, ServerWorks, Phoenix Technologies, VTM, Texas Instruments, Vital Technical Marketing.

Уставом PCI SIG предусматривается поддержка:

- прямой совместимости всех разработок локальной шины PCI или приложений на ее основе;
- спецификации локальной шины PCI как простой, легкой в использовании и устойчивой технологии;
- утверждению локальной шины PCI в качестве промышленного стандарта и технически долговечной архитектуры.

Членство в SIG доступно всем участникам компьютерного рынка. При этом членам предоставляются следующие преимущества:

- представление на рассмотрение изменений спецификации и приложений;
- участие в рассмотрении таких изменений;

- первоочередное получение изменений и приложений спецификации;
- право голоса при выборе членов комитета управления;
- присвоение идентификационных номеров (ID) производителям;
- техническая поддержка PCI;
- поддержка по работе с документацией и материалами по PCI;
- участие в программе встреч, конференций, спонсором которых является SIG, и других акциях, связанных с локальной шиной PCI.

За информацией относительно того, как стать членом SIG или для получения документации по локальной шине PCI, следует обращаться по адресу **www.pcisig.com**.

Спецификация моста PCI-to-PCI

Мост PCI-to-PCI соединяет две независимые шины PCI. Основная функция моста — обеспечение взаимодействия между мастером на одной шине PCI и *целевым устройством* (целью) на другой шине PCI. Мосты PCI-to-PCI позволяют системным и интегральным разработчикам расширять пределы допустимых электрических нагрузок, создавая иерархические шины PCI.

Спецификация для мостов PCI-to-PCI определяет требования к таким устройствам, устанавливая соответствие с основной спецификацией PCI. Спецификация не описывает детали реализации любого конкретного требования, а также не описывает конкретные реализации моста PCI-to-PCI.

Спецификация управления питанием PCI

Спецификация шины PCI представляет законченный документ с твердым определением протоколов, электрических характеристик и механических параметров. При этом в спецификации отсутствуют требования для поддержки функций управления питанием.

Управление питанием на современной платформе выполняется комбинацией BIOS и системного режима управления, иначе *SMM* (System Management Mode), использующими аппаратные средства, уникальные для каждой платформы. Не существует стандартного метода, чтобы точно установить, когда система занята и когда она находится в состоянии "свободна". Операционная система располагает этой информацией, так что разумно отдать ей функции управления питанием. Основной преградой для реализации данной идеи является недостаток стандартов для обеспечения операционной системы необходимой информацией. Ее наличие должно позволить управлять аппаратными средствами независимо от платформы. Спецификация "PCI Bus Power Management Interface" направлена на решение этой проблемы. Данная специ-

фикация определяет четыре состояния питания для шины PCI, четыре состояния питания для функций PCI, а также интерфейс для управления всеми состояниями питания.

Цель этой спецификации — установить стандартный комплект периферийного оборудования управления питанием PCI, интерфейсов аппаратных средств и протоколов. Такая инфраструктура позволяет операционной системе более четко управлять функциями питания шины PCI [6].

Основные задачи интерфейса управления питанием PCI:

- ❒ допустить разделение на уровни функций питания PCI;
- ❒ установить стандарт для функции PCI "пробуждающие события";
- ❒ установить стандарт для отчетности о состоянии управления питанием;
- ❒ установить стандартный механизм для управления состоянием питания шины PCI;
- ❒ обеспечить минимальное влияние на спецификацию шины PCI;
- ❒ обеспечить обратную совместимость со спецификацией шины PCI версии 2.2, исправленной версией 2.1 и доработками версии 2.0;
- ❒ сохранить для разработчиков возможности проектирования различных изделий;
- ❒ обеспечить единую архитектуру для всего рынка от мобильных приложений до серверов.

Спецификация "горячего" подключения PCI

Назначение данной спецификации — обеспечить бесперебойную работу файловых и прикладных серверов при *"Hot Plug"* ("горячее" подключение, без выключения питания) извлечении или замене плат расширения. Хотя эта разработка предназначалась для серверов, основные принципы применимы и для настольных ПК.

Существенные изменения в аппаратной части касаются платформ, а не плат расширения. Это сделано исключительно для ускорения рыночного признания подключения "Hot Plug". То есть спецификация описывает новый класс платформ, практически не затрагивая платы расширения. Не изменены и электрические требования для платы по сравнению с уже установленными (в спецификации шины PCI). Таким образом, существующие платы расширения будут обладать возможностью "горячего" подключения. При разработке предполагалось, что подавляющее большинство плат расширения PCI, разработанные до публикации PCI Hot Plug Specification, удовлетворят требованиям данной спецификации [8].

Основными задачами данного документа являются:

- определить условия для проектирования платы расширения, чтобы она соответствовала стандарту "Hot Plug";
- определить только необходимые характеристики для платформы "горячего" разъема так, чтобы это как можно меньше отразилось на разработчиках операционных систем;
- установить единую терминологию и архитектуру для "горячего" разъема системных аппаратных средств и программного обеспечения.

Особенно в спецификации "горячего" подключения подчеркиваются задачи, не относящиеся к данному документу:

- определить подробную реализацию платформы PCI;
- определить реализацию или структуру программных процедур "горячего" разъема. Представлены только понятия и характеристики, сама программная реализация строго определяется разработчиками операционных систем;
- определить поведение функций операционной системы более высокого уровня. Например, действия над файловой системой во время замены платы контроллера. Такие функции могут измениться от одной операционной системы к другой, и реализуются разработчиками операционной системы;
- определить функции "горячего" разъема на уровне драйверов. Например, равнозначная замена плат, дополнения платы новой версии. Аппаратные средства и программные компоненты низкого уровня, описанные здесь, требуются для всех систем "Hot Plug".

Спецификация Small PCI

Спецификация Small PCI (SPCI, спецификация PCI в миниатюрном исполнении) определяет альтернативную механическую реализацию 32-битной локальной шины PCI. Small PCI, прежде называвшаяся SFF PCI (Small Form-Factor PCI, маленькая PCI), предназначенная, в основном, для портативных компьютеров, логически совпадает с обычной шиной PCI.

Спецификация Small PCI использует тот же сигнальный протокол, электрические параметры, определения пространства конфигурации, что и спецификация шины PCI. При этом 64-битное расширение не предусматривается, и при частоте 33 МГц обеспечивается пропускная способность 132 Мбайт/с.

Для ссылок в описании протокола и пространства конфигурации следует обращаться к спецификации шины PCI. Спецификация Small PCI определяет аппаратную среду PCI. Как показал опыт, характеристики исполнения шины

PCI в настольной системе и системе сервера обусловили возможность использовать ее на меньших системных платформах [15].

Спецификация PCI BIOS

Этот документ описывает программный интерфейс, представляемый функциями BIOS. Интерфейс обеспечивает аппаратно-независимый метод управления устройствами PCI в компьютере. Назначение документа — изолировать разработчиков программного обеспечения от управления устройствами на аппаратном уровне. Таким образом, PCI BIOS является "надстройкой" над физическим уровнем с точки зрения управления.

Документ также предоставляет достаточно информации для системных разработчиков, чтобы создавать функции BIOS для конкретной системотехники (системной архитектуры, разработки).

Функции PCI BIOS предоставляют программный интерфейс к аппаратным средствам, используемым при разработке PCI-систем. Основное применение — осуществление различных операций в специфическом адресном пространстве PCI (пространство конфигурации и специальный цикл).

Функции PCI BIOS должны строиться таким образом, чтобы действовать во всех режимах архитектуры *x*86:

- режиме реального времени;
- защищенном режиме процессора 286;
- защищенном режиме процессора 80386;
- защищенном режиме 0:32 (также известном как режим, использующий плоскую модель памяти, в которой все сегменты начинаются по линейному адресу 0 и распределяют целое 4-гигабайтное адресное пространство).

Доступ к функциям PCI BIOS обеспечивается через прерывание INT 1Ah.

Для 32-битного режима (защищенный режим) доступ обеспечивается вызовом через точку входа: соответствующая функция кода PCI BIOS — B1h. Специфические функции BIOS вводятся, используя код подфункции. Пользователь устанавливает необходимые значения регистров процессора для вызываемой функции и подфункции и вызывает программное обеспечение PCI BIOS. Статус возвращается через *флаг переноса* (CF) и регистры процессора, специфические для введенной функции [4].

Спецификация мобильного PCI

Оригинальная спецификация PCI Mobile Design Guide была опубликована 27 октября 1994 года и предназначалась для управления питанием в системе. В то время мобильные платформы были основными для разработчиков.

Развитие концепции управления питанием устройств, которые находятся на шине PCI, проходило в направлении от мобильных платформ в настольные системы и далее в серверные. Чтобы сформулировать требования по управлению питанием, были разработаны две формальные спецификации: Advanced Configuration and Power Interface *(ACPI)* и PCI Power Management Interface Specification *(PCI-PM)*. ACPI была направлена на компоненты системной платы, в то время как PCI-PM — на рынок плат расширения. Спецификация ACPI требует, чтобы BIOS контролировал и управлял устройствами, в то время как спецификация PCI-PM определяет общий программный интерфейс в пространстве конфигурации PCI.

PCI Mobile Design Guide определяет малое время задержки, высокий уровень исполнения взаимосвязи шинных соединений, малое число контактов и интерфейс "склеивающей логики", идеальный для мобильных систем.

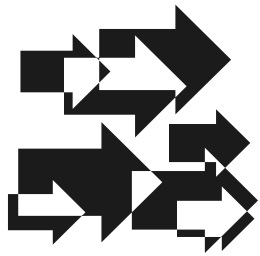
Данная спецификация является справочником для разработчиков в том случае, если мобильные системы и компоненты включают шину PCI. Руководство разработки предназначено облегчить разработку системы или компонентов, базовых для PCI [12].

Спецификация Mini PCI

Спецификация Mini PCI определяет платы расширения, эквивалентные обычным PCI-платам, но имеющим малые размеры: 69,9×46,0×5,6 мм. Предложенная компьютерными компаниями, включая 3Com, Compaq Computer, Dell Computer, Gateway, Hitachi PC, Toshiba и Intel, спецификация использует тот же сигнальный протокол, электрические характеристики и конфигурационное пространство, что и основная спецификация шины PCI. Отсутствие описания каких-либо параметров указывает на то, что они полностью совпадают с аналогичными в спецификации PCI. Основные особенности плат расширения Mini PCI:

- платы и разъемы Mini PCI обладают малым физическим размером;
- использование стандартных вспомогательных сигналов для сетевых и звуковых приложений;
- поддержка сигнала CLKRUN#, определенного в спецификации PCI Mobile Design Guide;
- отсутствует поддержка для дополнительных сигналов JTAG и для 64-разрядного расширения.

ГЛАВА 2



Элементная база шины PCI

Пользователям ПК достаточно знать, что существует системный блок и устройства ввода-вывода типа клавиатуры, монитора и мыши. Продвинутые пользователи знакомы с основными типами устройств и разъемов. Они мыслят категориями "процессор", "чипсет", "память", PCI и т. п. Чтобы понять работу шины PCI, а тем более разрабатывать устройства на ее основе, необходимо "опуститься" до уровня логических вентилей, операционных усилителей и транзисторов, и разобраться в схемотехнических особенностях реализации.

Схемотехнические основы построения шин

Прежде чем рассматривать электрические параметры и требования, предъявляемые спецификацией шины PCI к компонентам, системным платам и платам расширения, сделаем краткий экскурс в историю развития схемотехники. Еще в 1960-е годы появилась *резистивно-транзисторная логика* (РТЛ), позволяющая достаточно просто реализовать основные элементы вычислительной техники в виде набора повторяющихся стандартных РТЛ-модулей. Позже была разработана *диодно-транзисторная логика* (ДТЛ), а вслед за ней универсальная быстродействующая логика *SUHL* (Sylvania Universal High-level Logic) фирмы Sylvania, которая более известна как *транзисторно-транзисторная логика* (ТТЛ). Основными задачами при разработке первых элементов цифровой логики были повышение быстродействия и унификация модулей. В результате появились высокоскоростные ТТЛ-схемы, а также самая быстродействующая *эмиттерно-связанная логика (ЭСЛ)*. Унификация осуществлялась по двум направлениям. Во-первых, все сложные элементы составляли из однотипных ячеек, например, ТТЛ. Во-вторых, в пределах одной ячейки разработчики пытались уменьшить номенклатуру используемых элементов полупроводниковой технологии, заменяя, по возможности, транзи-

сторами все остальные компоненты (диоды, конденсаторы и резисторы). По мере усложнения реализуемых устройств актуальность приобрели вопросы уменьшения питающих напряжений и энергопотребления. Дальнейшее развитие микроэлектроники привело к тому, что в начале 1970-х годов были созданы первые *КМОП-элементы* (элементы с "комплиментарной металл-окисел-полупроводник" структурой). В то время они еще уступали по быстродействию элементам ТТЛ. Однако уже в 1980-е годы были разработаны *интегральные микросхемы* (ИМС) на основе КМОП-технологии с быстродействием и выходными параметрами, соответствующими ТТЛ. Рассмотрим основные преимущества и недостатки этих двух типов элементов.

Характеристики элементов ТТЛ и КМОП

Напряжение питания. Напряжение питания для элементов ТТЛ составляет +5 В ±5%, в то время как семейства КМОП имеют более широкий диапазон напряжений: от +2 до +6 В для микросхем серий HC, AC и от +3 до +15 В для серий 4000B, 74C. Семейства HCT и ACT, разработанные для совместной работы с биполярными ТТЛ, рассчитаны на напряжение питания +5 В.

Быстродействие и мощность. Биполярные ТТЛ-семейства потребляют значительный ток покоя — тем больший, чем быстрее семейство (AS и F). Их предельные рабочие частоты лежат в диапазоне от 25 МГц (для LS) до 100 МГц (для AS и F). Все семейства КМОП потребляют практически нулевой ток покоя. Однако их рассеиваемая мощность линейно возрастает с ростом частоты (требуется ток для переключения емкостной нагрузки). В результате элементы КМОП, работающие на наивысшей частоте, часто рассеивают такую же мощность, что и эквивалентные ТТЛ. Диапазон частот КМОП-элементов простирается от 2 МГц (для 4000B/74C при +5 В) до 100 МГц (для ACT/AC).

Входной каскад. Вход вентиля ТТЛ в состоянии низкого логического уровня представляет собой токовую нагрузку для управляющего им источника сигнала (типовое значение 0,25 мА для серии LS). Следовательно, для поддержания на входе вентиля низкого логического уровня необходимо обеспечить отвод тока. Поскольку выходные каскады схем ТТЛ обладают хорошей нагрузочной способностью, сопряжение друг с другом элементов ТТЛ не представляет проблемы. Она может возникнуть, если потребуется подключить входы ТТЛ к выходам микросхем другого типа. Вентили КМОП, напротив, практически не потребляют входной ток. Однако входы этих элементов чувствительны к статическому электричеству и могут выходить из строя при манипуляциях с ними.

В обоих семействах на неиспользуемые входы в зависимости от ситуации следует подавать высокий или низкий логический уровень.

Порог срабатывания. Порог срабатывания ТТЛ определяется падением напряжения на двух диодах по отношению к земле (порядка 1,3 В). Для элементов КМОП значение порога равно приблизительно половине напряжения питания, но может колебаться в широких пределах (типично от 1/3 до 2/3 напряжения питания). КМОП-семейства HCT и ACT спроектированы с низким значением порога для совместимости с элементами ТТЛ.

Выходной каскад. Выходной каскад вентиля ТТЛ в состоянии низкого логического уровня ведет себя как насыщенный транзистор, напряжение на котором близко к потенциалу земли (до 0,4 В), а в состоянии высокого логического уровня — как повторитель с высоким выходным напряжением, равным примерно напряжению питания U_+ минус падение напряжения на двух диодах. Для всех КМОП-семейств (включая HCT и ACT) выход представляет собой открытый полевой транзистор, подключенный к земле или шине питания. Обычно быстродействующие семейства (F, AS, AC, ACT) имеют более высокую нагрузочную способность, чем низкоскоростные (LS, 4000D, 74C, HC, HCT) [23].

Нагрузочная способность. В пределах одного логического семейства выходы элементов легко стыкуются с входами и не требуют решения вопросов сопряжения. Например, выходы элементов ТТЛ могут быть нагружены не менее чем на 10 входов ТТЛ (характеристика носит название *коэффициента разветвления по входу*: для ТТЛ его значение равно 10). Таким образом, при разработке устройств на ТТЛ-элементах обычно не возникает необходимость в использовании дополнительных согласующих схем. То же самое можно сказать и про элементы КМОП.

Шинная организация

Компьютер при грубом рассмотрении представляет собой систему, в которой несколько функциональных блоков должны обмениваться данными. Центральный процессор (ЦП), память и периферийные устройства должны иметь возможность передавать и получать, например, 32-разрядные слова. Если для соединения каждого устройства с каждым применить 32-жильный кабель, то возникшие расходы не будут оправданы производительностью. Для разрешения этой проблемы используется общая шина, доступная для всех устройств. При таком способе организации необходимо достичь соглашения о том, кому разрешено в данный момент передавать или получать данные. Отсюда вытекают такие термины, как арбитр шины, ведущий шины (мастер), ведомый (целевое устройство) и управление шиной [9, 23].

Для возбуждения шины нельзя использовать схемы с активным выходом, т. к. их нельзя отключить от общих информационных линий (в любой момент времени выходы таких устройств, подключенных к шине, будут находиться

в одном из двух возможных состояний — высокого или низкого логического уровня). В этом случае необходим вентиль, выход которого может находиться в "обрыве", т. е. быть отключенным от обеих линий питания. Такие устройства разработаны и имеют две разновидности, которые носят название "элементы с тремя состояниями" и "элементы с открытым коллектором".

Логика с тремя состояниями

Логические элементы с тремя состояниями, также называемые *TRI-STATE* (товарный знак National Semiconductors Corp., создавшей их), или тристабильными, представляют наиболее рациональное решение. На самом деле эти элементы не являются элементами с тремя уровнями напряжений. Это обычные логические схемы, которые имеют третье состояние выхода — *высокоимпедансное* ("обрыв"). Они имеют отдельный вход разрешения, с помощью которого могут быть переведены либо в состояние обычных активных выходов, либо в третье состояние независимо от других входных сигналов. Выходы с тремя состояниями имеются во многих типах ИМС: счетчиках, защелках, регистрах и т. п., а также в вентилях и инверторах. Устройство с выходом на 3 состояния функционирует подобно обычной логике с активным выходом, если подан сигнал разрешения. При этом на выходе существует либо высокий, либо низкий логический уровень. Если же сигнал разрешения отсутствует, схема отключает свой выход, и другие устройства могут использовать ту же самую линию.

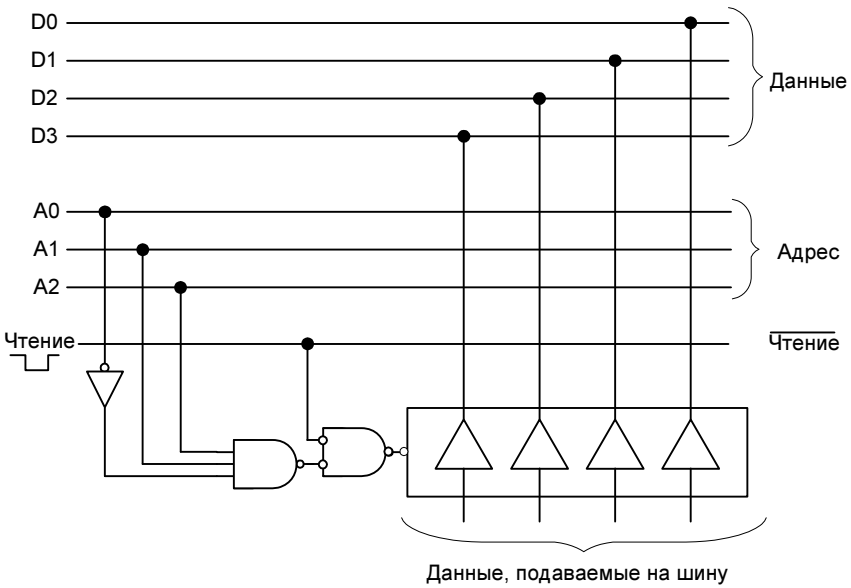


Рис. 2.1. Пример организации шины данных на элементах с тремя состояниями

Буферы ввода-вывода с тремя состояниями широко применяются для организации шин адресов и данных в компьютерах. Пример такого использования показан на рис. 2.1. Каждое устройство (память, периферия и т. п.), которому необходимо выставить данные на шину, связывается с ней через вентили с тремя состояниями (или через регистры). Только одно устройство выдает разрешение своим формирователям на работу с шиной. Все остальные устройства переводят шинные формирователи в третье состояние. Обычно выбранное устройство "узнает" о том, что оно должно выдавать данные на шину, опознав свой адрес на адресных и управляющих шинах [23].

Заметим, что должна существовать некоторая внешняя логика, гарантирующая, что устройства с тремя состояниями, подключенные к одним и тем же линиям шины, не будут пытаться передавать сигналы одновременно. С этой точки зрения идеальным решением является назначение каждому устройству собственного уникального адреса.

Логика с открытым коллектором

Предшественником логики с тремя состояниями была логика с открытым коллектором. Она позволяет подключиться к одиночной линии нескольким формирователям. Выход с открытым коллектором просто не содержит активной нагрузки у транзистора выходного каскада, как показано на рис. 2.2 (здесь приведен вариант *ТТЛ-элемента с диодами Шотки*, сокращенно *ТТЛШ*).

Применение элементов логики с открытым коллектором подразумевает установку внешнего нагрузочного резистора между источником питания и открытым коллектором выходного транзистора. При малых значениях сопротивления этого резистора обеспечивается относительно большое быстродействие и помехоустойчивость, однако растет рассеиваемая мощность и нагрузочный ток выходного каскада. Для ТТЛ-элементов типичными являются значения в пределах от нескольких сотен до нескольких тысяч Ом. Вариант организации шины на основе логики с открытым коллектором можно получить, если на рис. 2.1 заменить приемопередатчики с тремя состояниями на двухвходовые вентили И-НЕ с открытым коллектором, подключив один вход каждого вентиля к сигналу (высокого логического уровня), разрешающему подключение к шине. Заметим, что данные на шине при таком включении будут инвертированы. Каждую линию шины необходимо через нагрузочный резистор подключить к источнику питания с напряжением +5 В.

К недостаткам логики с открытым коллектором следует отнести пониженные быстродействие и помехоустойчивость по сравнению с обычными схемами, использующими активную нагрузку. Вот почему формирователи с тремя состояниями являются основными элементами для реализации шин в компью-

терах. Однако существуют три ситуации, в которых требуется устройство с открытым коллектором:

- управление внешними нагрузками;
- организация "проводного ИЛИ";
- внешние шины [23, 24].

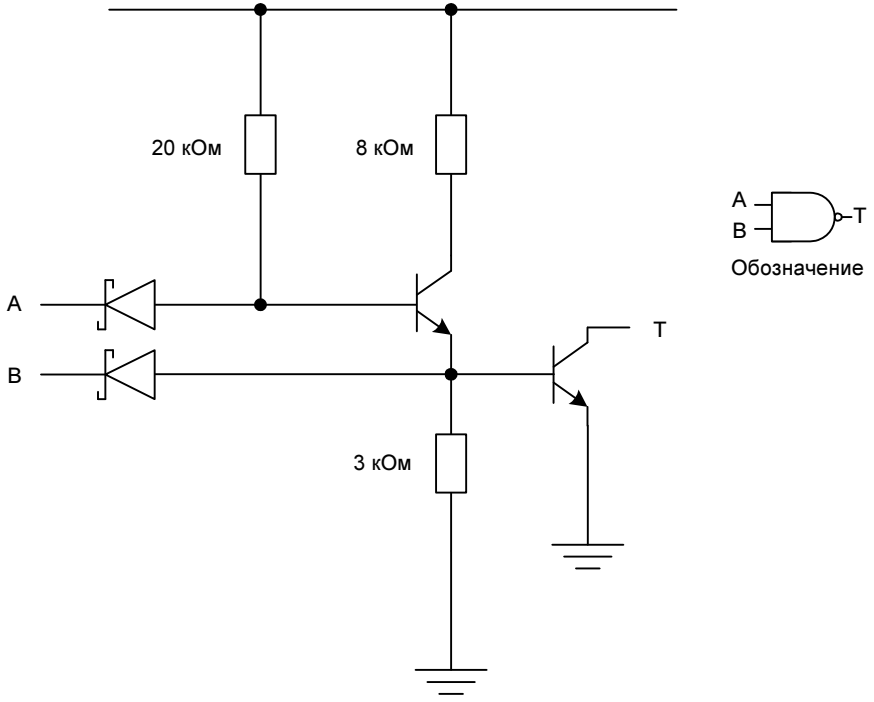


Рис. 2.2. ТТЛШ-вентиль И-НЕ с открытым коллектором

Управление внешней нагрузкой. Логика с открытым коллектором хорошо приспособлена для управления внешней нагрузкой, которая подключается к источнику положительного напряжения. В частности, можно включить маломощную 12-вольтовую лампочку или сформировать логический перепад 15 В с помощью резистора, установленного между выходом вентиля и источником +15 В (рис. 2.3).

Проводное ИЛИ. Если объединить вместе несколько вентилей с открытым коллектором, как это показано на рис. 2.4, то получится так называемая схема *"Проводное ИЛИ"*, — соединение, которое ведет себя подобно большому вентилю ИЛИ-НЕ, выдающему на выходе низкий логический уровень, если какой-либо вход имеет высокий уровень.

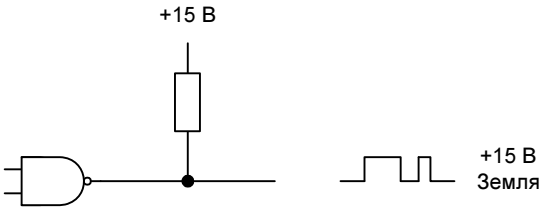


Рис. 2.3. Управление внешней нагрузкой

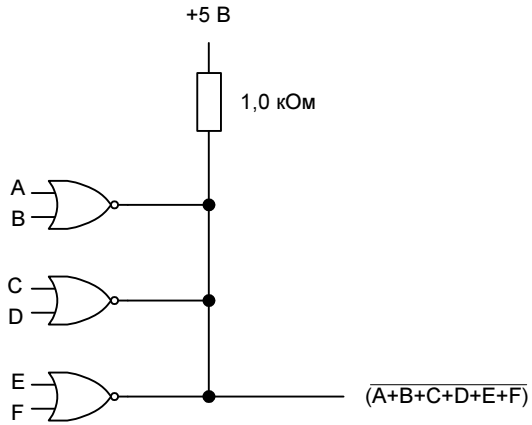


Рис. 2.4. Проводное ИЛИ

Такое объединение недопустимо при использовании схем с активной нагрузкой: если между всеми вентилями не будет согласовано, каким должен быть выходной сигнал, то возникнет режим конкуренции. Объединять можно схемы ИЛИ-НЕ, И-НЕ и т. п. Это соединение также иногда называют "Проводное И", поскольку высокий уровень на его выходе возникает лишь тогда, когда он существует на выходе каждого вентиля (состояние разомкнутого, или открытого выхода). Оба этих названия описывают одну и ту же схему, которая представляет собой "Проводное И" при положительной логике и "Проводное ИЛИ" — при отрицательной. "Проводное ИЛИ" пользовалось популярностью в ранние дни цифровой электроники. Сегодня оно применяется довольно редко за двумя исключениями:

1. В логических семействах ЭСЛ (выходы которых можно назвать "открытый эмиттер"), элементы могут свободно объединяться по схеме "Проводное ИЛИ".
2. Существуют особые линии (например, линия прерывания в компьютерных шинах), функциями которых является не передача информационных раз-

рядов, а просто индикация того, что хотя бы одно устройство требует внимания. В этом случае используется "Проводное ИЛИ", поскольку оно дает необходимую функциональность и не требует дополнительной внешней логики.

Внешние шины. В применениях, где быстродействие не является определяющим фактором, можно встретить формирователи с открытым коллектором, используемые для возбуждения шин. Наиболее частый случай — это выдача данных из компьютера. Общими примерами являются шины, применяемые для связи компьютера с дисководом, и инструментальная шина стандарта IEEE-480, известная как *GPIB* (General-Purpose Interface Bus, интерфейсная шина общего назначения) [23].

Проблемы разработки цифровых устройств

При разработке цифровых устройств на базе элементов КМОП и ТТЛ могут возникать проблемы общего характера. Рассмотрим некоторые из них.

Тупиковое состояние в статическом режиме

Допустим, имеется какое-то устройство с рядом триггеров, которые в процессе работы проходят через заданные состояния. Схема функционирует без ошибок и сбоев, но вдруг замирает после возникновения определенных обстоятельств. Единственный способ заставить ее вновь заработать — это выключить питание, а затем снова его включить. Такая ситуация может возникнуть из-за того, что схема имела *мертвое состояние* (запрещенное состояние системы, которого не удалось избежать). В это состояние устройство могло попасть, например, под воздействием коммутационных помех в цепи питания. При разработке цифровых схем очень важно выявить подобные состояния (путем анализа соответствующего конечного автомата) и строить логику таким образом, чтобы схема могла автоматически восстанавливаться. Как минимум, должен быть предусмотрен сигнал начальной установки (вырабатываемый при нажатии на специальную кнопку, при включении питания и т. д.), который мог бы возвращать систему в нормальное состояние. При наличии такого сигнала никаких других мер может и не потребоваться. Как правило, это сигнал системного сброса [23].

Начальная установка

Всякая система должна иметь начальное состояние и сигнал, переводящий ее в такое состояние. Иначе при включении питания в системе могут возникать непредсказуемые ситуации. На рис. 2.5 приведен пример формирования сигнала сброса [23].

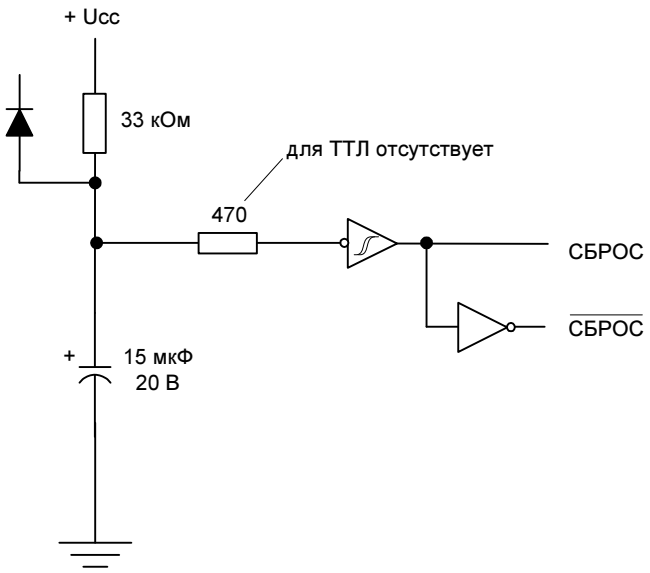


Рис. 2.5. Схема формирования сигнала сброса

Резистор 470 Ом, подключенный последовательно к входу вентиля, необходим в случае использования КМОП-схем. Он позволяет избежать повреждения схемы при отключении питания, т. к. в противном случае электролитический конденсатор будет пытаться запитать систему через защитный диод входного вентиля КМОП. Хорошей идеей является применение *триггера Шмитта*, благодаря которому снятие сигнала СБРОС происходит быстро. Символ гистерезиса на рисунке означает, что на входе инвертора установлен триггер Шмитта, собранный, например, на микросхемах ТТЛ-серии 74LS14 (6 инверторов), либо КМОП 40106, либо 74C14 [23].

Переключения

Иногда вентили управляются сигналами от триггеров. Необходимо убедиться в том, что в схеме не может возникнуть ситуация, в которой к моменту тактирования триггера вентиль открывается, но по истечении задержки на триггере — закрывается (и в результате, на выход проходят "иголки"). Кроме того, сигналы данных, возникающие на входах триггеров, не должны быть задержаны по отношению к тактовым импульсам. В общем, должны задерживаться такты, но не информация [23].

Метастабильные состояния

Триггер и любое тактируемое устройство могут сбиться, если изменение сигналов на информационных входах произойдет непосредственно перед воз-

никновением тактового импульса (менее чем за время установки $t_{уст}$). В худшем случае выходной сигнал триггера будет буквально совершать колебания в окрестностях порога срабатывания в течение нескольких микросекунд (для сравнения, нормальная величина задержки распространения элементов ТТЛ составляет 20 нс). Разработчики логических схем обычно не принимают это во внимание, но подобная проблема может возникнуть в быстродействующих системах, когда потребуется синхронизировать асинхронные сигналы. В этом случае требуется лишь установить цепочку синхронизаторов или "детектор метастабильных состояний", который будет сбрасывать триггер [23].

Крутизна фронтов тактовых импульсов

Крутизна фронтов тактовых импульсов оказывает большее влияние на схемы КМОП, чем на ТТЛ. Проблема возникает в том случае, когда для тактирования нескольких соединенных между собой устройств используется сигнал с большим временем нарастания, как показано на рис. 2.6.

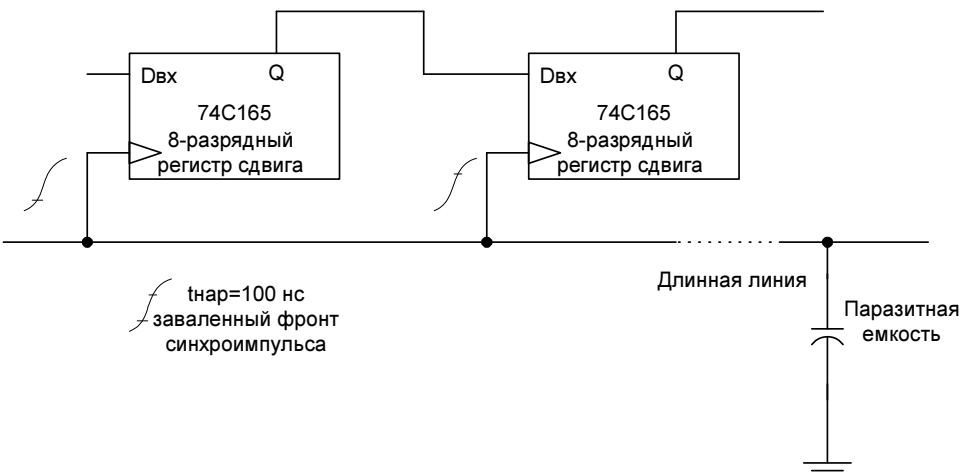


Рис. 2.6. Перекос тактовых импульсов

В рассматриваемом примере два регистра сдвига тактируются по фронту с большим временем нарастания. Это время обусловлено емкостной нагрузкой выхода КМОП, который имеет относительно высокий импеданс (порядка 500 Ом при работе от источника +5 В). Проблема возникает из-за того, что порог срабатывания у первого регистра может оказаться ниже, чем у второго. В результате сдвиг первого регистра произойдет раньше второго, и последний бит первого регистра будет потерян. Дело еще осложняется тем, что значения пороговых напряжений для КМОП-устройств колеблются в очень широком диапазоне (фактически они могут принимать любое значение в преде-

лах от 1/3 до 2/3 напряжения питания V_{cc}). В подобной ситуации самое лучшее — это расположить корпуса микросхем рядом, избегая тем самым большой емкостной нагрузки по тактовым входам.

Короче говоря, тактовые входы каких-либо цифровых микросхем должны всегда тщательно обрабатываться. Например, тактовые линии с шумом или "звоном" должны всегда очищаться с помощью вентиля (возможно с входным гистерезисом) до подачи на синхронизируемую микросхему. Например, медленная логика 4000B или 74C, питающая быстрые семейства HC или AC, наверняка вызовет проблемы перекоса импульсов или кратных переходов [23].

Укороченные импульсы

Когда счетчики (по модулю n) должны сбрасываться собственным входным сигналом, необходимо ввести задержку для того, чтобы предотвратить появление укороченного импульса. То же самое относится и к импульсам записи в счетчики или регистры сдвига. Укороченные импульсы часто приводят схему к работе на границе устойчивости и вызывают периодические сбои. При разработке схемы следует исходить из наихудшего значения для задержки [23].

Электронные компоненты шины PCI

Основные преимущества КМОП-элементов, такие как высокое быстродействие, малые стоимость и потребление, обусловливают их применение в ПК по сей день.

Переход к напряжению 3,3 В

Начиная со спецификации версии 2.2 разработчики стандарта локальной шины PCI осуществили постепенный переход от сигнальной среды с напряжением 5 В к 3,3 В. Например, стандарты PCI-X, Mini PCI и PCI 66 поддерживают только сигнальную среду 3,3 В. Спецификация PCI версии 2.3 определяет два типа разъемов для плат расширения: один для сигнальной среды 3,3 В и один для 5 В. Каждому типу разъема соответствует свой тип платы расширения, как это показано на рис. 2.7. Поддержка систем с сигнальной средой 5 В сохраняется только в целях обратной совместимости. Плата расширения для среды 3,3 В не может быть вставлена в разъем для среды 5 В, и наоборот. Это достигается за счет разных физических размеров и форм разъемов в указанных средах.

Тип сигнальной среды определяется системной платой. Плата расширения для среды 3,3 В может работать только в своей среде. Универсальная плата расширения предназначена для функционирования в обеих сигнальных сре-

дах. Она способна определить тип сигнальной среды и адаптироваться к ней. Оба типа плат расширения могут содержать компоненты, работающие от источника напряжения 3,3 и/или 5 В. Различие между этими типами заключается в используемом сигнальном протоколе, а не во внутренних шинах питания и не в технологии изготовления компонентов [9].

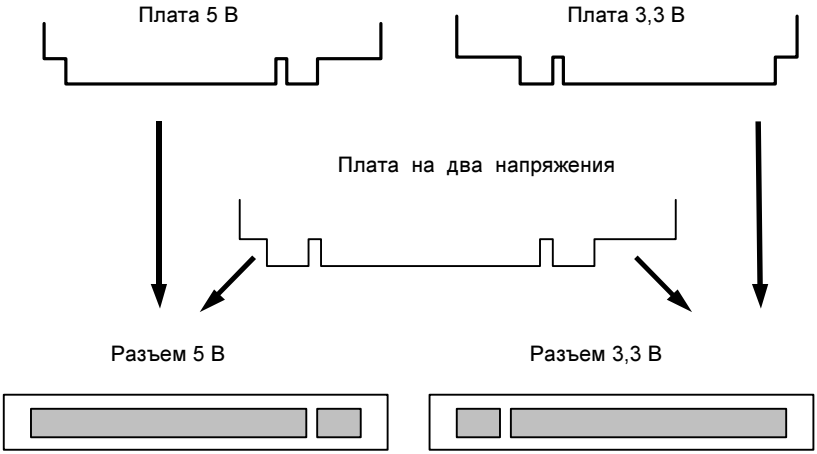


Рис. 2.7. Разъемы плат расширения

Буферы PCI-компонентов на универсальной плате расширения должны быть совместимы с обоими напряжениями сигнальных сред. Существует множество буферных реализаций, которые могут соответствовать обеим средам. В их числе буферы с двойными напряжениями, способные работать с любой из двух шин питания. В сигнальной среде 3,3 В буферы платы подключены к шине 3,3 В. Когда та же самая плата расширения установлена в разъем 5 В, эти буферы подключаются к шине 5 В. В результате, обеспечивается совместимость универсальной платы расширения с любой сигнальной средой. Самое главное, что все платы расширения должны поддерживать сигнальную среду 3,3 В. Универсальные платы расширения нужны для систем, в которых реализованы только разъемы 5 В [9].

Характеристики буфера ввода-вывода

Шина PCI имеет два свойства, которые предопределяют различные подходы к выбору характеристик буфера ввода-вывода. Во-первых, шина PCI ориентирована на технологию КМОП. Это означает, что токи установившегося *статического режима*, возникающего после того, как закончились переходные процессы, будут минимальны. Фактически основным потребляемым то-

ком при этом становится ток утечки в высокоомных "подтягивающих" резисторах, привязывающих уровень сигнала к уровню напряжения питания. Во-вторых, работа шины PCI основана на *отраженной волне*, что позволяет добиться более высокого быстродействия по сравнению с вариантом использования прямой волны. Это значит, что шинные формирователи выдают лишь половину требуемого напряжения переключения (естественно, за меньшее время). Электрическая волна распространяется вдоль шины, отражается от ее нетерминированного конца без изменения знака и амплитуды, и возвращается в обратном направлении. Таким образом, удваивается начальный перепад напряжения, достигая требуемого уровня. Шинные приемники находятся в середине диапазона переключения в течение всего времени распространения волны. Процесс длится до 10 нс, что составляет треть времени цикла шины 33 МГц. Буферы ввода-вывода тратят эту относительно большую часть времени на переключения, находясь в переходном, *динамическом режиме*. При этом постоянная составляющая потребляемого тока минимальна, так что типовой метод определения характеристик буфера, основанный на токах утечки, не эффективен. В этом случае более подходящим будет использование характеристик переменного тока. Основное значение приобретают *вольт-амперные характеристики* (ВАХ) приемников для активного диапазона переходных процессов. ВАХ подбирают так, чтобы достичь приемлемого поведения буфера в режиме переключений для двух крайних (по нагрузке) конфигураций: четыре нагрузки на системной плате и два разъема для плат расширения, или две нагрузки на системной плате и четыре разъема для плат расширения. Естественно, что на практике допускаются и другие конфигурации, учитывающие фактическое использование оборудования, способы размещения, полное входное сопротивление при нормальной нагрузке системной платы и т. д. [9, 24, 23].

Мост PCI-to-PCI

Связь одной шины с другими осуществляется посредством *мостов*. В современных материнских платах основные функции мостов реализованы в чипсетах, но для специфических применений мосты могут быть выполнены и в виде отдельных микросхем [13].

Назначение и терминология

Мост PCI-to-PCI обеспечивает соединение между двумя независимыми шинами PCI. Основная функция моста состоит в том, чтобы разрешить транзакции между мастером на одной шине PCI и целевым устройством на другой шине PCI. Мосты PCI-to-PCI предоставляют возможность преодолеть огра-

ничения, обусловленные электрической нагрузкой, путем создания иерархических шин PCI.

> ***Примечание***
>
> Основным механизмом передачи данных на шине PCI является блочный механизм. Блок состоит из фазы адреса и одной или более фаз данных. Совокупность фазы адреса и одной или нескольких фаз данных называется *транзакцией*. Транзакция происходит между мастером и целью, в направлении от мастера. *Мастером* называется логический узел устройства, выполняющий функции генерирования транзакций, а *целью* — логический узел устройства, выполняющий функции конечного объекта транзакции.

Для дальнейшего рассмотрения протокола и разделов, прямо или косвенно связанных с мостом PCI-to-PCI, необходимо ввести некоторые определения.

- *Мост (bridge)* — в спецификации слово мост означает мост PCI-to-PCI. Другие типы мостов именуются полностью.
- *Первичный или основной интерфейс (primary interface)* — PCI-интерфейс моста, который соединяется с шиной PCI, расположенной со стороны главного моста.
- *Вторичный интерфейс (secondary interface)* — PCI-интерфейс моста, находящийся с противоположной стороны от первичного интерфейса. Вторичный интерфейс соединяется с шиной PCI, которая находится за мостом. То есть между этой шиной и процессором располагается главный мост и мост PCI-to-PCI.
- *Нисходящий поток (downstream)* — транзакции, которые перенаправляются из первичного интерфейса моста во вторичный.
- *Исходная шина (originating bus)* — шина, на которой находится мастер транзакции, проходящей через мост.
- *Шина назначения (destination bus)* — это шина, на которой расположено целевое устройство транзакции, проходящей через мост.
- *Восходящий поток (upstream)* — транзакции, передаваемые из вторичного интерфейса в основной.

Таким образом, мост имеет два PCI-интерфейса, основной и вторичный. Каждый интерфейс поддерживает операции целевого устройства и мастера. Функции моста как целевого устройства на исходной шине — выступать от имени ведомого, который фактически размещен на шине назначения. Функции моста как мастера на шине назначения — выступать от имени мастера, который фактически размещен на исходной шине.

На рис. 2.8 показаны два типичных применения моста. В первом мост используется для создания дополнительного сегмента шины PCI с разъемами. Этот сегмент обозначен на рисунке как шина PCI 1. В этом примере основной

интерфейс моста 1 соединен с шиной PCI 0, вторичный интерфейс соединен с шиной PCI 1. Второй пример состоит в использовании моста для создания сегмента шины PCI на плате расширения, которая допускает размещение на ней нескольких устройств PCI. В этом примере основной интерфейс моста 2 соединен с шиной PCI 1, а его вторичный интерфейс соединен шиной с PCI 2. Номер, назначенный мосту, соответствует номеру сегмента, порожденного мостом. Главный мост рассматривается под номером 0, поскольку порождает сегмент шины PCI 0.

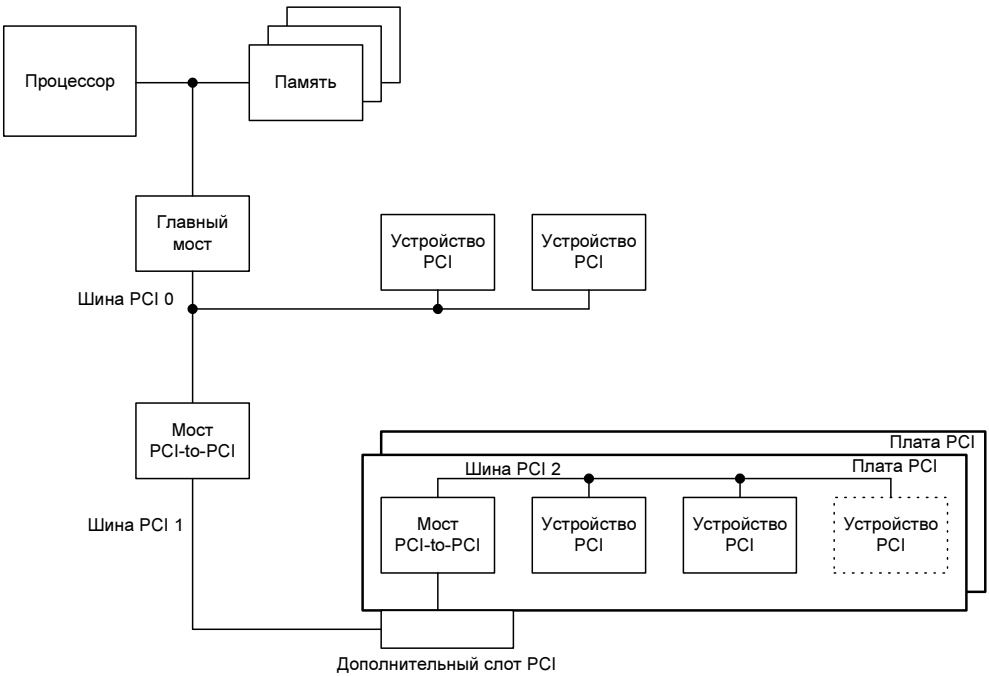


Рис. 2.8. Применение моста PCI-to-PCI

Мост разрешает выполнение транзакций между мастером на одном PCI-интерфейсе и целевым устройством (целью) на другом, как показано на рис. 2.9. Интерфейс цели на одной шине соединен с интерфейсом мастера на другой. Блоки моста на пути данных между основным и вторичным интерфейсом обеспечивают любую необходимую буферизацию адреса и данных транзакции. Блок цели, соединенный с основным интерфейсом PCI, должен поддерживать конфигурационное пространство PCI. Мост по существу состоит из четырех конечных автоматов — двух мастеров и двух целей. Каждый из конечных автоматов интерфейса мастера или цели должен удовлетворять требованиям спецификации *"PCI Local Bus Specification"* [13].

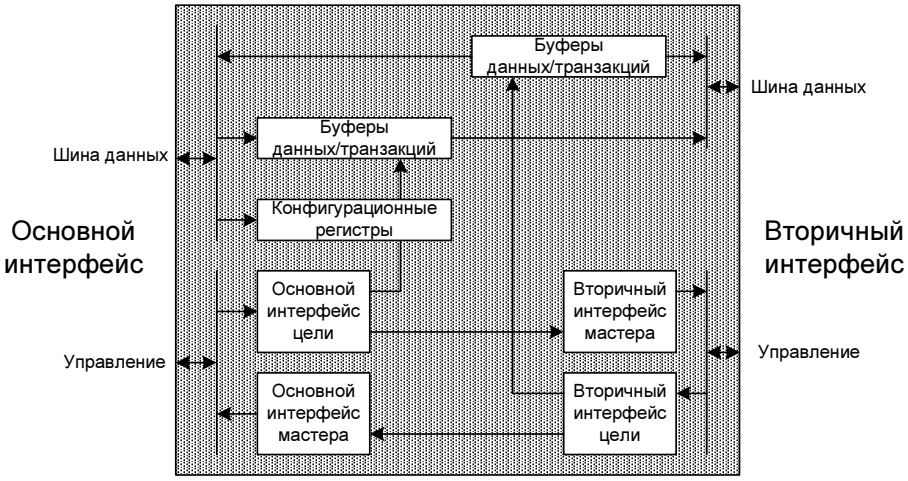


Рис. 2.9. Пример структуры моста PCI-to-PCI

Примеры мостов

Примером моста PCI-to-PCI является мост Advanced PCI Bridge (APB, продвинутый мост PCI). Эта микросхема совместима со спецификацией PCI. APB обеспечивает соединительный путь между 32-битной шиной, функционирующей на частотах до 66 МГц на основном интерфейсе, и двумя 32-битными шинами 5 или 3,3 В, работающими на частотах до 33 МГц, на вторичном интерфейсе. В основном он предназначен для систем, построенных на базе процессоров UltraSPARC-IIi. Мост APB обеспечивает этот микропроцессор прямым доступом к устройствам, присоединенным к шине PCI и отображенным в пространство ввода-вывода, с минимальным временем задержек. В дополнение, он обеспечивает мастера PCI прямым, высокопроизводительным доступом к основной памяти. Использование APB зависит от организации шины PCI.

Еще одним примером служит мост PCI-to-ISA фирмы Winbond W83628F. Интегральная микросхема (ИМС) W83629D является усеченным вариантом центральной микросхемы для управления *запросами на прерывания* (Interrupt Request, IRQ) и *прямым доступом к памяти* (Direct Memory Access, DMA). Микросхемы W83628F и W83629D вместе образуют полный набор для моста PCI-to-ISA. В современных поколениях чипсетов поддержка шины и слотов ISA отсутствует. Мост рассматривается лишь в качестве примера. В свое время указанные микросхемы обеспечивали лучшее комплектное решение для чипсета не-ISA. Также форматы корпуса W83628F (128-QFP) и W83629D (48-LQFP) были выбраны в большинстве экономичных решений для снижения их стоимости и сохранения параметров размещения.

Операционные усилители

Понятие "обратная связь" широко распространено и выходит за рамки какой-либо узкой области техники. В системах управления обратная связь используется для сравнения выходного сигнала с заданным значением и выполнения соответствующей коррекции. В усилительной схеме (линейной) выходной сигнал пропорционален входному, поэтому в усилителе с обратной связью входной сигнал сравнивается с определенной частью выходного сигнала. *Отрицательная обратная связь* — это процесс передачи выходного сигнала обратно на вход, при котором погашается часть выходного сигнала. Отрицательная обратная связь уменьшает коэффициент усиления, но при этом она улучшает другие параметры схемы, например, устраняет искажения и нелинейность, сглаживает частотную характеристику (приводит ее в соответствие с заданной), делает поведение схемы предсказуемым. Чем глубже отрицательная обратная связь, тем меньше внешние характеристики усилителя зависят от характеристик усилителя с разомкнутой обратной связью (ОС), и, в конечном счете, оказывается, что они зависят только от свойств самой схемы ОС. Операционные усилители обычно используют в режиме глубокой обратной связи, а коэффициент усиления по напряжению в разомкнутой петле ОС достигает в этих схемах миллиона. *Операционный усилитель* (ОУ) — это дифференциальный усилитель постоянного тока с очень большим коэффициентом усиления и несимметричным выходом.

На ОУ строятся самые разнообразные схемы:

- ❒ инвертирующий усилитель;
- ❒ неинвертирующий усилитель;
- ❒ усилитель переменного тока;
- ❒ повторитель;
- ❒ источники тока;
- ❒ линейные схемы:
  - схема с инвертированием по выбору;
  - повторитель со следящей связью;
  - идеальный преобразователь тока в напряжение;
  - дифференциальный усилитель;
  - суммирующий усилитель;
  - усилитель мощности (бустер);
  - источник питания;

- ❒ нелинейные схемы:
  - усилитель с переключением мощности;
  - активный выпрямитель.

В качестве примера можно привести микросхему, применяемую в некоторых материнских платах — LM358. Это устройство состоит из двух независимых операционных усилителей, разработанных для функционирования от одного источника в широком диапазоне напряжений. Функционирование от расщепленных источников также возможно, если различия между двумя источниками составляет от 3 до 30 В, и значение напряжения Vcc составляет по меньшей мере 1,5 В. Применения данной микросхемы включают усилители-преобразователи, блоки усиления постоянного тока, и все обычные схемы операционных усилителей, которые могут быть реализованы в системах с одним питающим напряжением. Для примера, эти устройства могут функционировать от стандартного источника с напряжением 5 В в цифровых системах и в состоянии обеспечить требуемый интерфейс без дополнительных пятивольтовых источников [23].

Обзор технологий CompactPCI и PCI-X

Рассмотрим более подробно два особых варианта локальной шины PCI. Первый из них, магистрально-модульный стандарт CompactPCI, ориентирован на системы специального назначения (автоматизация промышленности, военное дело). Он применяется, прежде всего, во встраиваемых компьютерах. Второй стандарт PCI-X ориентирован на более высокую производительность.

CompactPCI

В 1995 году в рамках международной *Ассоциации PICMG* (PCI Industrial Computer Manufacturers Group, группа производителей промышленных компьютеров на базе PCI) была сформирована отдельная рабочая группа, состоящая из специалистов ведущих американских компаний. Она должна была изучить возможности использования PCI в качестве системной широковещательной шины, пригодной для создания широкого класса надежных промышленных и коммуникационных встраиваемых систем.

Разработка CompactPCI должна была удовлетворять следующим требованиям:

- ❒ поддерживать не менее 8 слотов расширения на пассивной объединительной магистрали;
- ❒ обеспечивать полную логическую и электрическую совместимость с PCI;

- ❒ эффективно использовать стандартные многозадачные операционные системы типа Windows NT, Solaris, OS 9, QNX, VxWorks, LynxOS и т. п.;
- ❒ применять стандартный промышленный евроконструктив 3U (100×160 мм) и 6U (233×160 мм).
- ❒ использовать широкодоступные, недорогие стандартные полупроводниковые PCI-компоненты;
- ❒ обеспечивать эффективное конвекционное и/или кондукционное охлаждение модулей;
- ❒ иметь возможность использования ключевых мезонинных технологий для организации гибкого ввода-вывода (расширения системы посредством дополнительных шин и стандартных низкопрофильных промышленных модулей *(мезонинов)*, размещаемых на платах-носителях и образующих вместе с ними типовые комплекты индустриального стандарта, например, IndustryPack (IP));
- ❒ обеспечивать максимально простую интеграцию со стандартными шинами типа VMEbus;
- ❒ иметь возможность "горячей замены" модулей без выключения системы.

В результате в 1995 году была представлена спецификация стандарта "CompactPCI" версии 1.0 [1, 2].

Ключевые особенности стандарта CompactPCI

Стандарт CompactPCI (CPCI) электрически и логически совместим со спецификацией PCI. Шина CompactPCI является мультиплексированной, синхронной, процессорно-независимой, ее разрядность составляет 32 или 64 бита. Связь между различными сегментами шины организована через мосты PCI-to-PCI. Скорость передачи данных составляет 132/264 Мбайт/с (для шины 32/64 бита). Алгоритм арбитража является централизованным и одноуровневым. Шина строится на основе технологии КМОП 5 В или 3,3 В [2].

Шина CompactPCI допускает использование любых типовых компонентов обычной PCI. Основой стандарта является общепринятая промышленная технология создания высоконадежных встраиваемых систем — пассивная объединительная магистраль.

Стандарт поддерживает и включает следующие функции и возможности:

- ❒ распространенный механический формат — европлаты формата 3U и 6U;
- ❒ дополнительные функции ввода-вывода через вспомогательные задние разъемы;
- ❒ полную процессорную и программную независимость. Шина может применяться для таких процессорных архитектур, как PowerPC, Alpha, Pentium *x*86, AMD, SPARC, MC68360 и т. д.;

- независимость от операционных систем (ОС). Может использоваться любая ОС, как общего назначения типа Windows 2000, Linux, так и ОС реального времени, например, VxWorks, LynxOS, QNX;
- поддержку технологии Plug & Play;
- использование стандартного разъема повышенной надежности на 235 контактов, расположенных в 5 рядов с шагом 2 мм.

Благодаря новым качествам системного разъема стало достижимым двойное количество слотов расширения по сравнению с обычным стандартом PCI.

Стандарт CompactPCI распространился чрезвычайно быстро. Номенклатура имеющихся сегодня на рынке изделий позволяет говорить о широком использовании его в самых разных системах [1, 2].

Процессорные модули

Важнейшим элементом любой встраиваемой системы является процессорный модуль (обычно, одноплатный). Имеющиеся на рынке системы CompactPCI форматов 3U и 6U построены на базе процессоров PowerPC, Pentium, AMD, MC68360, UltraSPARC и DSP. Данные системы являются компьютерами повышенной надежности с расширенными функциями промышленных систем [2].

Системы ввода-вывода

Успеху стандарта способствовала возможность обеспечить функции ввода-вывода для огромного числа применений.

Стандарт CompactPCI унаследовал некоторые особенности промышленных магистрально-модульных систем (ММС): модульность, механический евроконструктив 3U или 6U, простоту и скорость интеграции стандартных систем ввода-вывода, высокую надежность и компактность, простоту обслуживания.

CompactPCI также унаследовал лучшие черты настольных систем: недорогую и быструю элементную базу PCI, технологию Plug & Play, совместимую с широчайшим спектром программного обеспечения, особенно для платформ семейства MS Windows.

Благодаря тому, что концепция CompactPCI переняла все лучшее от шины PCI, новые устройства быстро нашли применение. С помощью мезонинных технологий IP и PMC они охватили широкий спектр систем: от промышленного контроллера до коммуникационного сервера. Вот несколько примеров.

Компания SBS Green Spring Computers (США) выпустила плату-носитель PCI-40 (PCI плата-носитель четырех модулей IP) в формате 6U CompactPCI (cPCI200) и в формате 3U CompactPCI (cPCI100). Работа была выполнена в

кооперации с фирмами FORCE и ProLog. Выпуск IP-носителей CompactPCI-200/100 означал, что компаниям-сборщикам (*OEM*, Original Equipment Manufacture, изготовитель комплектного оборудования) и системным интеграторам предоставляется доступ к большому количеству объектов ввода-вывода. Системы ввода-вывода всегда были представлены достаточно широко: аналоговые и цифровые, графические и телекоммуникационные; под промышленные, локальные и глобальные сети; для управления двигателями/приводами, реле; со встроенными энергонезависимой памятью, таймерами, синтезаторами речи, функциональными генераторами и т. д.

Компании CES и MOTOROLA выпустили платы-носители PMC (PCI Mezzanine Card, плата для мезонинных модулей на базе шины PCI, стандарт IEEE1386) формата CompactPCI 6U. В связи с этим выпуском для компаний-сборщиков и системных интеграторов открылся новый рынок PMC-мезонинных модулей широкой номенклатуры, позволяющий гибко компоновать на платах высокопроизводительные расширения.

В 1997 на рынок вышли со своей серийной продукцией CompactPCI традиционные крупные производители открытых промышленных ММС и коммуникационных систем: MOTOROLA, SUN, FORCE, PEP Modular Computers, SBS Green Spring Computers и многие другие [2].

Производители

Фирма PEP Modular Computers специализируется на производстве малогабаритной продукции CompactPCI формата 3U для встраиваемых промышленных систем на базе Pentium.

Фирма MOTOROLA является безусловным мировым лидером в производстве аппаратуры CompactPCI, в том числе систем формата 6U с процессорами PowerPC, 6U PC/AT-совместимых компьютеров на основе Pentium, и базовых механических платформ CompactPCI формата 6U. Решения от MOTOROLA ориентированы на использование широкого класса современных операционных систем типа MS Windows и стандартных операционных систем реального времени: OS9, VxWorks, QNX, LynxOS, pSOS+. Высокое качество производства этой компании дает возможность предоставить потребителям пятилетнюю гарантию на продукцию CompactPCI [2].

SBS Green Spring Computers — это традиционный лидер в производстве стандартных, платформо-независимых мезонинных модулей ввода-вывода (IndustryPack, PC-MIP и PMC), обеспечивает более 12960000 разнообразных решений только на европлатах CompactPCI формата 6U.

Совместимые с IndustryPack и PMC мезонинные модули компаний N.A.T., Condor, ACTIS, LSI, TEWS, Pentland и др. существенно расширяют возможности системных интеграторов и сборщиков по организации гибкого ввода-вывода.

Большой спрос на устройства CompactPCI инициировал возникновение новых фирм-производителей. Одной из них является западно-германская компания INOVA Computers. Специалисты и менеджеры компании смогли добиться эффективных ценовых характеристик производимых модульных компонентов 3U CompactPCI при соответствующих технических достоинствах. Решения компании INOVA Computers, несмотря на возможность применения широкого класса операционных систем, адаптированы для использования стандартной ОС MS Windows с расширениями реального времени [2].

Шина PCI-X

Архитектура PCI была испытана временем и доказала свою применимость. За период существования PCI было выпущено большое количество устройств и разъемов. Однако появились новые высокопроизводительные системы ввода-вывода, которые по своим возможностям превосходили характеристики, определяемые стандартом PCI. Идеальным решением для них был бы стандарт, отвечающий новым требованиям производительности при поддержке обратной совместимости с предыдущими поколениями PCI.

В соответствии с данными требованиями был разработан стандарт PCI-X 2.0. Он поддерживает две частоты: 266 МГц (PCI-X 266) и 533 МГц (PCI-X 533). Получаемая в результате полоса пропускания в два и в четыре раза больше полосы пропускания шины PCI-X 133 и в 32 раза больше исходной полосы пропускания PCI.

PCI-X 2.0 сохраняет многие элементы из предыдущих поколений PCI и, следовательно, наследует опыт разработок. Операционные системы, разъемы, драйверы устройств, форм-факторы, протоколы, *BIOS* (Basic Input/Output System, базовая система ввода-вывода), сигнальная среда, *BFM* (Bus Functional Model, функциональная модель шины), и другие элементы, учитываемые в различных версиях спецификаций шины PCI, также учитываются в спецификации PCI-X 2.0. Многие модели этих элементов остались неизменными. Такая преемственность снижает временные и материальные затраты на разработку, причем сохраняется накопленный инженерами опыт. В результате уменьшается время выхода на рынок, риск новой технологии. Переход на PCI-X 2.0 осуществлялся в несколько этапов из-за большого количества плат и адаптеров PCI на рынке [14].

Общие сведения

Известно, что шина PCI предусматривает тактовые частоты 33 и 66 МГц. При 64-разрядной шине это позволяет получить пропускную способность 533 Мбайт/с. Дальнейшему повышению частоты препятствует следующее требование стандарта: периферийное устройство должно декодировать полу-

ченный сигнал в течение того же такта, на котором происходит отправка, а на следующем такте выдать ответ. При частоте 66 МГц для этой операции остается всего 3 нс. В спецификации PCI-X эта проблема решается с помощью *межрегистрового протокола*. В соответствии с ним сигнал отправителя удерживается на протяжении следующего такта специальным триггерным регистром, предоставляя устройству полный такт для декодирования. Это позволяет повысить частоту тактирования шины до 133 МГц и при размерности в 64 разряда достичь скорости 1 Гбайт/с. Кроме увеличения тактовой частоты, повышение производительности достигается с помощью таких усовершенствований протокола, как введение новой фазы транзакции, механизма расщепленных транзакций и ряда других. Атрибутивная фаза использует 36-битное поле, которое описывает транзакцию на шине более детально, чем традиционный протокол PCI. В частности, оно включает информацию об объеме и порядке транзакции, идентификаторе устройства-инициатора транзакции. Все вместе это позволяет снизить требования к порядку выполнения транзакций, применить более эффективные алгоритмы управления буфером и повысить степень утилизации шины и остальных системных ресурсов.

Суть *механизма расщепленных транзакций* заключается в следующем. В соответствии с протоколом PCI устройство, запросившее данные, определяет момент их готовности путем постоянного опроса источника, что по существу приостанавливает работу шины. Новый протокол устраняет фазу опроса. Согласно ему, устройство-инициатор посылает источнику данных только сигнал запроса, получая в ответ квитанцию. После этого связь между устройствами прекращается, и регистр освобождается для обработки другой информации. Когда данные будут готовы, источник инициирует новую транзакцию и пошлет их запросившему устройству.

Остановимся на отдельных характеристиках архитектуры PCI-X, делающих ее весьма привлекательной как для производителей, так и для пользователей ПК. Прежде всего, необходимо отметить, что спецификация поддерживает до 256 шинных сегментов, каждый из которых инициализируется независимо от других. Основной ее особенностью является обратная совместимость с существующими шинами, адаптерами, устройствами и драйверами спецификации PCI. Поэтому могут существовать как однородные системы, содержащие только адаптеры PCI-X, так и смешанные, включающие также и адаптеры PCI. Следует сказать, что максимальная частота шины 133 МГц поддерживается лишь в том случае, когда сегмент содержит только один разъем. При двух разъемах она снижается до 100 МГц, а при четырех — до 66 МГц [14].

Первым документом, определяющим стандарт PCI-X, является *спецификация "PCI-X Addendum to the PCI Local Bus Specification"* ("Приложение PCI-X к спецификации локальной шины PCI"). Эта спецификация была надстройкой над комплектом спецификаций PCI, в который входит спецификация шины

PCI, спецификация моста PCI-to-PCI, спецификация управления питанием и спецификация PCI hot-plug ("горячее" подключение устройств к шине без выключения питания системы) [14].

Первоначально PCI-X определяла следующие расширения и дополнения к спецификации PCI, необходимые для увеличения скорости передачи данных и повышения эффективности использования шины:

1. Тактовая частота до 133 МГц.
2. Введение в протокол межрегистрового обмена. Данное изменение позволяет снизить количество тактов, устройство должно ответить на любое входное изменение в пределах двух тактов.
3. Добавление служебной информации в каждую транзакцию, что обеспечивает более эффективное использование схемы управления буферами:
   - каждая транзакция в последовательности указывает полное количество байтов для чтения или записи. Если транзакция разорвана, то новая транзакция, которая продолжит последовательность, содержит обновленное значение счетчика байтов;
   - каждая транзакция включает идентификатор инициатора и последовательность транзакции (или "нить"), к которой она принадлежит. Идентификатор содержит номер шины, номер устройства и номер функции;
   - дополнительная информация о порядке следования транзакции и требованиях кэша.
4. Оптимизация правил разрыва соединения и введение ограничений на использование состояний ожидания для более эффективного использования ресурсов шины и памяти:
   - инициатору запрещено вставлять состояния ожидания;
   - целевым устройствам запрещено вставлять состояния ожидания после начальной фазы данных;
   - ведущим и ведомым устройствам разрешено заканчивать блочную транзакцию только на естественно выровненной 128-байтной границе. Это требование устанавливает нижний предел размера блока для более эффективного использования ресурсов главной шины и памяти. Ведомым устройствам разрешено разрывать транзакцию только после одной фазы данных в адресном диапазоне, не поддерживающем транзакции большого размера.
5. Задержанные транзакции шины PCI заменены расщепленными транзакциями на шине PCI-X. Все транзакции, кроме транзакций записи в память, должны быть выполнены немедленно или через протокол расщепленных транзакций. По протоколу расщепленных транзакций целевое устройство

завершает транзакцию сообщением "Split Response" (расщепленный ответ), выполняет команду и инициирует транзакцию "Split Completion" (расщепленное завершение) для передачи данных или сообщения инициатору.

6. Более широкие возможности по восстановлению ошибок для устройств шины PCI-X, за счет чего снижено вмешательство системы при ошибках четности данных.

Большинство требований к PCI-X определены в соответствии с требованиями к PCI. Это упростило задачу преобразования разработки PCI в PCI-X и изготовление устройств типа PCI-X, функционирующих в среде PCI. Следующие параметры PCI-X совпадают с параметрами PCI:

- ❒ устройства имеют 32- или 64-битную шину данных;
- ❒ адрес и данные мультиплексированы на одних линиях шины;
- ❒ транзакции содержат одну или две адресные фазы, одну или больше фаз данных;
- ❒ устройства дешифрируют адрес и команду и выставляют сигнал DEVSEL# для подтверждения транзакции;
- ❒ механическая спецификация платы расширения;
- ❒ наименования сигналов и выводов разъема (кроме одного нового вывода, PCIXCAP);
- ❒ напряжения питания;
- ❒ максимальное потребление питания для плат расширения;
- ❒ один тактовый сигнал для всех устройств и транзакций. Изменения состояния шины и передача данных производится по переднему фронту синхроимпульса;
- ❒ "горячее" подключение — "Hot Plug" и последовательность "горячей" замены — "Hot Swap".

Следующие параметры PCI-X в целом совпадают с PCI, но имеют незначительные отличия:

- ❒ протокол обмена управляется сигналами FRAME#, IRDY#, DEVSEL#, TRDY#, и STOP#;
- ❒ фазы данных выполняются при выставленных сигналах IRDY# и TRDY#. Для PCI-X определены дополнительные правила обмена целевого устройства (цели);
- ❒ правила следования транзакций и передача транзакций мостами. Задержанные транзакции PCI заменены расщепленными транзакциями в PCI-X;

- уровни напряжений в основном не изменены. Значение V_{il} (max) увеличено в режиме PCI-X для увеличения предела допустимого шума.

Ключевые особенности

- *Обеспечение требуемой полосы пропускания*. PCI-X 2.0 определяет приемлемую полосу пропускания для поддержки большинства передовых технологий, таких как 10-Gigabit Ethernet, 10-Gigabit FibreChannel. Допускается поддержка нескольких высокоскоростных портов, например, плата PCI-X 533 поддерживает два порта 10-Gigabit Ethernet или 10-Gigabit FibreChannel.
- *Полная аппаратная обратная совместимость*. Одним из факторов успеха стандарта PCI является поддержка обратной совместимости каждого поколения. PCI-X 2.0 обеспечивает полную обратную совместимость на аппаратном уровне вплоть до плат адаптеров PCI 32-бит/33 МГц. Платы расширения для среды 3,3 В или универсальные могут быть вставлены в слот PCI-X 2.0 и наоборот, платы расширения PCI-X 2.0 могут быть вставлены в слот PCI.
- *Полная программная обратная совместимость*. Переход в среду PCI-X 2.0 также обусловливается полной обратной совместимостью на программном уровне. Операционные системы, поддерживающие все современные поколения PCI, например, Windows 98/2000/XP, Linux, OpenVMS, Novell, UNIX, OS/2, SCO, Nonstop Computing и др. поддерживают и PCI-X 2.0. Вся адресация, регистры и протоколы поддержаны и интерпретируются правильно для PCI-X 2.0. Программная совместимость выполняется для BIOS и драйверов устройств. Не требуется производить изменения в коде драйверов или BIOS, по умолчанию они совместимы со всеми частотами плат PCI-X 2.0.
- *Предназначение стандарта PCI-X 2.0*. Технология PCI-X 2.0 применима к любой компьютерной платформе, тем не менее, ее истинное назначение заключается в применении в высокопроизводительных системах типа промышленных серверов, профессиональных рабочих станций, мощных UNIX-серверов, мэйнфреймов, сетевых и коммуникационных систем [14].

Элементы PCI-X 1.0 и PCI, усиленные в PCI-X 2.0

Спецификация PCI-X 2.0 использует элементы предыдущих поколений обычной шины PCI и PCI-X 1.0. Большое количество ранее разработанных элементов значительно уменьшают время разработки продуктов и выхода их на рынок, при этом уменьшается риск невостребованности или краха разработки.

- *Архитектура PCI-X 2.0*. Одинаковый состав управляющих сигналов, которые поддерживают одинаковые функции для всех поколений.

- *Конечные автоматы*. Для реализации PCI-X 266 и PCI-X 533 требуются незначительные изменения в конечных автоматах. Это является одним из ключевых элементов разработки, облегчающим переход.
- *BFM* (Bus Functional Model). Спецификации PCI-X 266 и PCI-X 533 используют модель шины BFM, которая почти идентична модели для PCI-X 133. Изменились только несколько строк кода для создания BFM с новыми частотами.
- *Драйверы устройств*. Драйверы устройств PCI-X 266 и PCI-X 533 идентичны драйверам PCI-X 133 и PCI-X 66. Они могут быть использованы без какой-либо модификации [14].

Новые функциональные возможности

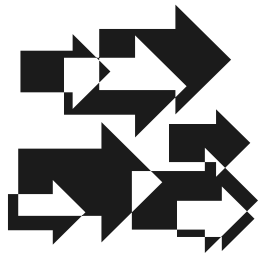
- *Стробы*. В спецификацию PCI-X 266 и PCI-X 533 добавлены стробы для гарантии передачи данных. Стробы используются для запуска входного такта фиксирования данных и гарантии фиксирования данных за определенное время.
- *Сигнальная среда 1,5 В*. Применение высоких частот требует снижения амплитуды сигнала. Поэтому для стандартов PCI-X 266 и PCI-X 533 предусмотрена сигнальная среда 1,5 В. Буферы ввода-вывода будут поддерживать оба сигнальных уровня для поддержки обратной совместимости с технологиями 3,3 В PCI.
- *Поддержка ECC*. Спецификация PCI-X 2.0 определяет дополнительную функцию для повышения помехоустойчивости — *ECC* (Error-Correcting Code, код с исправлением ошибок). Так как протокол использует для проверки восьмые биты, то он имеет возможность исправлять однобитовые ошибки и обнаруживать двухбитовые ошибки. Однобитовые ошибки исправляются автоматически, двухбитовые ошибки не исправляются и маркируются для повторной передачи. Введение ECC не влечет за собой снижения производительности; для ее реализации требуется такое же время, как и для обнаружения ошибок четности предыдущих поколений PCI.
- *Новые регистры спецификации PCI-X 2.0*. Для поддержки функциональности ECC в PCI-X 2.0 определены новые конфигурационные регистры.
- *Высокая эффективность шины*. Анализ предыдущих поколений PCI показал неэффективность протокола. Произведенные доработки протокола позволяют передавать данные не просто на более высокой частоте, но и гораздо более эффективно [14].

Заключение

Стандарт PCI имеет продолжительную для технологии и успешную историю, начавшуюся как высокоскоростной для своего времени стандарт 33 МГц. За

прошедшее время стандарт развился до сверхвысоких частот, нескольких скоростей передачи данных и эффективных шин. Каждый новый стандарт PCI был основан на предыдущем поколении. Залогом успеха PCI было обеспечение аппаратной и программной совместимости на каждом этапе. Разработка и реализация каждого поколения PCI требовала меньших временных и материальных затрат за счет обобщения предыдущих разработок и инженерного опыта. Спецификация PCI-X 2.0 является следующим этапом в этом наследии и предлагает производительность, необходимую для современных систем [14].

ГЛАВА 3



Интерфейс шины

Физический уровень шины PCI определяет типы поддерживаемых сигналов, количество линий, необходимых для организации взаимодействия устройств. В *разд. "Сигналы и линии шины"* подробно рассматривается каждая составляющая интерфейса.

Шина PCI определяет три физических адресных пространства:

- пространство ввода-вывода;
- адресное пространство памяти;
- конфигурационное пространство.

Пространства адресов памяти и ввода-вывода объединены, т. е. пространство ввода-вывода является частью пространства памяти. Порядок доступа и правила дешифрации различаются для каждого адресного пространства. В *разд. "Правила и порядок адресации"* определены типы поддерживаемых адресных пространств, методы дешифрации каждого из них. Описано назначение и условия формирования, передачи и исполнения команд шины.

Сигналы и линии шины

Реализация всех функций интерфейса PCI требует определенного количества линий. Необходимо выработать оптимальное соотношение числа выводов как для внешних связей, так и для межкомпонентных. По возможности разработчики свели количество линий к минимуму, реализовав все необходимые функции. Слишком большое число линий повлечет за собой увеличение размеров плат расширения, слишком малое может привести к нежелательным перегрузкам и снижению производительности. При этом хорошо бы учесть будущее развитие, повышение требований к пропускной способности (следо-

вательно, разрядности). Все перечисленное отразилось на интерфейсе в ходе разработки [9, 18].

Типы сигналов

Предваряя описание линий и сигналов PCI, необходимо сказать, что понимается под сигналами в спецификации. В отечественной теории электрорадиоцепей и сигналов (сокращенно ТЭРЦ) под *сигналом* понимается значение напряжения или тока. В спецификации часто происходит подмена понятий. С одной стороны, сигнал — это название линии (отдельного вывода), с другой стороны, сигнал — это передаваемый по линии импульс или пачка импульсов с определенными характеристиками. Таким образом, понятие "сигнал" смешано, в одном случае это физический вывод, в другом значение напряжения. Поэтому в дальнейшем под сигналом будем понимать его определение из ТЭРЦ, при этом описание сигнала подразумевает закрепление за ним вывода слота или устройства. Каждому выводу соответствует один сигнал, за исключением мультиплексированных линий.

Под *управлением сигналом* понимается способность агента переводить его в одно из допустимых состояний. *Агент* — любое устройство, подсоединенное к шине. Агент может быть как ведущим, так и ведомым устройством (ведущее устройство называется мастером, подчиненное или ведомое — *целевым устройством* или *целью*).

Интерфейс PCI использует 47 выводов для ведомого устройства, 49 для мастера. Такое количество выводов необходимо для реализации функций обработки данных, адресации, управления интерфейсом, арбитража и некоторых системных функций. На рис. 3.1 показаны сигналы, сгруппированные по функциональному назначению: слева указаны необходимые сигналы, а справа — опциональные. Для указания отличий сигналов мастера и цели представлено два рисунка, рис. 3.1 и 3.2. [9, 18]

Далее даны определения типов сигналов, общие для всех устройств на шине PCI. Исключение составляют арбитр и центральный ресурс. Термин *"центральный ресурс"* используется для описания основных функций шины, обычно обеспечиваемых в системе PCI мостом, либо стандартным интерфейсом. Для арбитра сигнал REQ# является входным, GNT# — выходным, остальные сигналы PCI для арбитра аналогичны сигналам для мастера или цели. Центральный ресурс является логическим устройством, где представлены все типы системных функций.

- ***in*** — *Input* — входной сигнал.
- ***out*** — *Totem Pole Output* — каскадный выход.
- ***t/s*** — *Tri-State* — двунаправленный вывод с тремя состояниями. Может использоваться как входной или выходной.

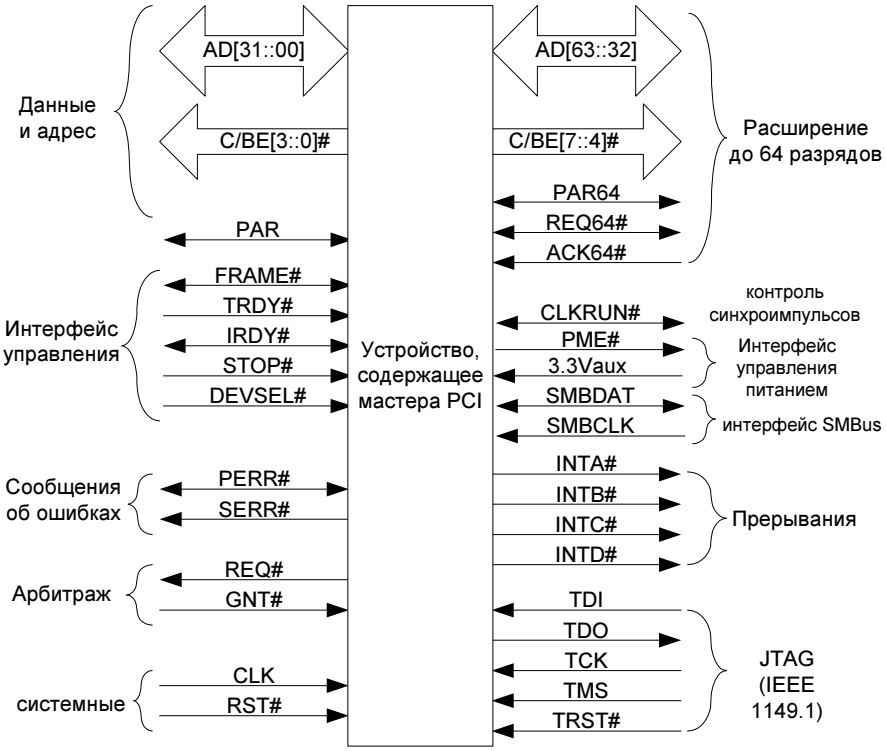


Рис. 3.1. Список сигналов PCI-мастера

- ***s/t/s*** — *Sustained Tri-State* — активный низкий сигнал с тремя состояниями. На отдельно взятом временном интервале должен управляться только одним агентом. Перед тем, как оставить сигнал в свободном состоянии, управляющий агент должен перевести сигнал в высокое логическое состояние. Другой агент может начать управлять сигналом s/t/s только через один такт после того, как предыдущий "владелец" сигнала переведет его в свободное состояние. Повышение уровня требуется для поддержания неактивного состояния, пока другой агент не начнет управлять сигналом. Неактивное состояние обеспечивается центральным ресурсом.
- ***o/d*** — *Open Drain* — открытый коллектор; позволяет использовать различное количество устройств путем объединения их с помощью схемы "монтажное ИЛИ". Повышение уровня требуется для поддержания неактивного состояния, пока другой агент не начнет управлять сигналом. Неактивное состояние также обеспечивается центральным ресурсом.

Описания линий и соответствующих им сигналов разбиты на функциональные группы в соответствии с рис. 3.1 и 3.2. Символ "#" в конце наименования сигнала показывает, что при низком логическом уровне напряжения сигнал

находится в активном состоянии. Если символ "#" отсутствует, активному состоянию сигнала соответствует высокий логический уровень напряжения. Фигурной скобкой на рисунках показано назначение выделенной группы сигналов, также показан тип сигнала, используемый для каждого вывода [9, 18].

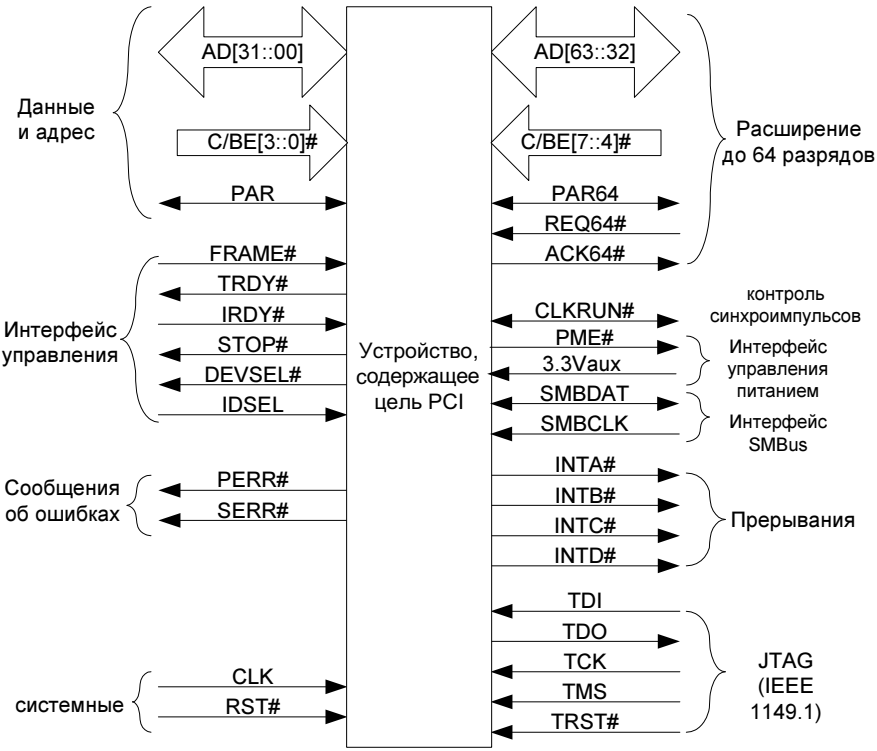


Рис. 3.2. Список сигналов PCI-цели

Обязательные линии

Системные выводы

- **CLK** (in). *Clock* обеспечивает синхронизацию всех транзакций на шине PCI, а также является входным для каждого PCI-устройства. Все сигналы PCI, за исключением RST#, INTA#, INTB#, INTC#, INTD#, PME# и CLKRUN#, являются дискретными по фронту CLK; временные параметры определяются относительно данной границы.
- **RST#** (in). *Reset* используется для приведения специфических регистров PCI, генераторов последовательностей и сигналов к соответствующему состоянию. В спецификации не описывается, что происходит с устройст-

вом при выставлении сигнала RST#, если PCI-секвенсор не поддерживает спецификацию PCI. Исключение составляют начальные состояния регистров PCI, которые необходимы для конфигурации.

Некоторые устройства могут генерировать "пробуждающие" события, пока шина находится в режиме пониженного потребления питания. В этом случае относительно линии RST# предъявляются дополнительные требования.

Во время нахождения сигнала RST# в активном состоянии все выходные сигналы должны находиться в третьем состоянии. Сигналы REQ# и GNT# должны также находиться в третьем состоянии (они не могут управляться низким или высоким уровнем во время сброса). Управление линиями AD, C/BE# и PAR во время действия сигнала RST# выполняет центральный ресурс (операция называется "парковка шины"), причем только по низкому логическому уровню, т. к. эти сигналы не могут управляться в высоком логическом уровне.

Активное и неактивное состояние сигнала RST# асинхронно по отношению к сигналу CLK. Несмотря на асинхронность, приведение сигнала к неактивному состоянию гарантируется для "чистого" фронта, свободного от биений (искажений). Исключение составляет случай, когда требуется доступ в пространство конфигурации. Это вызвано тем, что после инициализации могут "откликаться" только те устройства, которым требуется перезагрузить систему[9].

Выводы адреса и данных

- **AD[31::00]** (t/s). *Address Data* — адрес и данные мультиплексированы на одних и тех же выводах PCI. Шина PCI поддерживает поблочное чтение и запись. Транзакция шины состоит из фазы адреса[1], сопровождаемой одним или большим количеством фаз данных. Фаза адреса — это временной цикл, в котором активен сигнал FRAME#. В течение фазы адреса на линиях AD[31::00] содержится физический адрес (32 бита). При операциях ввода-вывода эти линии содержат адрес байта, при обращении к пространству конфигурации и памяти — адрес двойного слова (DWORD). Когда следуют фазы данных, линии AD[07::00] содержат младший значащий байт (lsb — low significant byte), а на линиях AD[31::24] содержится старший значащий байт (msb — most significant byte). Устойчивость записываемых данных обеспечивается активностью сигнала IRDY#, а устойчи-

[1] DAC — двойной цикл адреса, используется две фазы адреса для передачи 64-разрядного адреса.

вость читаемых данных — активностью TRDY#. Данные передаются только при условии одновременной активности сигналов IRDY# и TRDY#.

- **C/BE[3::0]** (t/s). Выводы *Bus Command* и *Byte Enables* ("команда шины" и "разрешение обращения к байтам") мультиплексированы на одних и тех же линиях PCI. Во время фазы адреса линии C/BE[3::0]# определяют команду шины. В течение фазы данных линии C/BE[3::0]# используются в качестве разрешения обращения к байтам *(Byte Enable)*. Сигнал действителен для всей фазы данных и определяет, какой или какие байты несут значимые данные. C/BE[0]# применяется к байту 0 (lsb), а C/BE [3]# применяется к байту 3 (msb).
- **PAR** (t/s). *Parity* — это сигнал контроля по четности[2] для линий AD[31::00] и C/BE[3::0]#. Генерирование кода контроля по четности требуется для всех агентов PCI. Сигнал PAR стабилен и допустим в течение одного такта после фазы адреса. Для фаз данных PAR допустим в течение такта после того, как будет активен IRDY# — при транзакции записи, или TRDY# — при транзакции чтения. Если выставлен только сигнал PAR, то он имеет значение в течение одного такта после завершения текущей фазы данных. (PAR имеет ту же самую синхронизацию, что и AD[31::00], но с задержкой на один такт.) Во время фаз данных и адреса при операциях записи управление сигналом PAR выполняется мастером; ведомое устройство управляет сигналом PAR для фаз данных при чтении.

Интерфейсные управляющие выводы

- **FRAME#** (s/t/s). *Cycle Frame* (циклический временной интервал). Управляется текущим мастером для указания начала и продолжительности доступа. Сигнал FRAME# выставляется, когда необходимо указать начало транзакции. Передача данных происходит, пока сигнал FRAME# находится в активном состоянии. Когда сигнал FRAME# становится неактивным, транзакция переходит в заключительную фазу данных или завершается.
- **IRDY#** (s/t/s). *Initiator Ready* (готовность инициатора) показывает готовность агента инициатора (мастера шины) завершить текущую фазу данных. IRDY# используется совместно с TRDY#. Фаза данных может завершиться в любой момент времени, пока сигналы IRDY# и TRDY# находятся в активном состоянии. Во время записи наличие сигнала IRDY# указывает, что на линиях AD[31::00] присутствуют достоверные данные. При чтении это означает, что мастер готов к приему данных. Циклы ожидания вставляются до тех пор, пока активны сигналы IRDY# и TRDY#.

---

[2] Сумма "1"-ц на линиях AD[31::00], C/BE[3::0]# и PAR должно быть равно четному числу.

- **TRDY#** (s/t/s). *Target Ready* (готовность целевого устройства) показывает готовность агента (выбранного устройства) завершить текущую фазу данных транзакций. Сигнал TRDY# используется совместно с IRDY#. Фаза данных может завершиться в любом такте во время нахождения в активном состоянии сигналов TRDY# и IRDY#. При чтении TRDY# указывает, что на линиях AD[31::00] присутствуют достоверные данные. Во время записи его активность означает готовность целевого устройства к принятию данных. Циклы ожидания вставляются до тех пор, пока активны оба сигнала IRDY# и TRDY#.
- **STOP#** (s/t/s). *Stop* (стоп) показывает, что ведомое устройство посылает мастеру запрос на остановку текущей транзакции.
- **LOCK#** (s/t/s). *Lock* (блокировать) показывает неделимую операцию для моста, для выполнения которой требуется несколько транзакций. При выставленном сигнале LOCK# могут выполняться неразделяемые (непрерываемые, заблокированные) транзакции к незаблокированному мосту. Разрешение выполнения транзакции на шине PCI не означает, что при этом контролируется сигнал LOCK#. Контроль над сигналом LOCK# можно получить по определенным правилам и при наличии сигнала GNT#. В то время как один мастер управляет выводом LOCK#, возможно использование шины PCI различными агентами. Блокированные транзакции могут быть инициированы только главными мостами, мостами PCI-PCI и мостами расширения.
- **IDSEL** (in). *Initialization Device Select* (выбор устройства при инициализации) используется для выбора конкретной микросхемы при операциях чтения и записи пространства конфигурации.
- **DEVSEL#** (s/t/s). *Device Select* (устройство выбрано), активное состояние сигнала показывает, что управляющее устройство дешифрировало данный адрес как цель текущего доступа. DEVSEL# в качестве входа показывает, было ли выбрано на шине какое-то устройство [9].

Арбитражные выводы

- **REQ#** (t/s). Сигнал *Request* (запрос) показывает арбитру, что данному агенту требуется доступ к шине. Сигнал относится к типу "от точки-к-точке". Каждый мастер должен иметь свой собственный вывод REQ#, и удерживать его в третьем состоянии, пока выставлен сигнал RST#.
- **GNT#** (t/s). Сигнал *Grant* (предоставление) показывает агенту, что предоставлен (разрешен) доступ к шине. Сигнал также относится к типу "от точки-к-точке". Каждый мастер должен иметь свой собственный вывод GNT#. Сигнал GNT# должен игнорироваться, пока выставлен сигнал RST#.

Пока сигнал RST# находится в активном состоянии, арбитр должен игнорировать все сигналы, приходящие по линии REQ#[3]. Это объясняется тем, что выводы REQ# находятся в третьем состоянии и не содержат запроса. Арбитр сможет выполнить арбитраж только после того, как сигнал RST# будет переведен в неактивное состояние. Мастер должен игнорировать сигнал на входе линии GNT#, пока выставлен RST#. Сигналы REQ# и GNT# были определены как тристабильные из-за требований согласования питания, т. к. возможна ситуация, когда арбитр шины и мастер шины питаются от различных источников питания.

Арбитражные выводы должны быть реализованы только в устройствах, содержащих мастера шины [9].

Выводы для сообщения об ошибках

Все устройства должны[4] содержать выводы сообщений об ошибках. Для реализации механизма обнаружения и восстановления ошибок четности используются следующие сигналы:

- **PERR#** (s/t/s). *Parity Error* (ошибка контроля по четности) предназначен для сообщения об ошибках контроля по четности во время всех транзакций PCI, за исключением специального цикла. Сигнал PERR# должен находиться в активном состоянии под управлением агента, получающим данные, в течение двух тактов после того, как обнаружена ошибка контроля данных по четности. Минимальная длительность сигнала PERR# — один такт для любой фазы данных, в которой обнаружена ошибка контроля данных по четности (если идут последовательно несколько фаз данных, в каждой из которых есть ошибка контроля данных по четности, то сигнал PERR# будет установлен более чем для одного такта). За один такт до того, как вывод PERR# перейдет в третье состояние со всеми соответствующими s/t/s-сигналами, он должен быть переведен в высокий логический уровень.
- **SERR#** (o/d). *System Error* (системная ошибка) предназначен для выдачи сообщений об ошибках контроля по четности адреса, ошибках контроля по четности данных для команды *Special Cycle*, или любых других системных ошибках, последствия которых могут оказаться фатальными. Возможны ситуации, когда агенту не требуется генерирование немаскируемого прерывания (NMI), следовательно, необходим механизм для сообщения о различных событиях. SERR# представляет собой вывод типа "открытый коллектор" и выставляется в течение одного такта PCI, когда агент сооб-

[3] REQ# входной для арбитра, GNT# — выходной.

[4] Для некоторых устройств системной платы допускается исключение.

щает об ошибке. Выставление SERR# синхронизировано по времени, при этом требуется время на установку и удержание всех сигналов на шине. Однако установка SERR# в неактивное состояние происходит при небольшом повышении уровня напряжения (до той же величины, что и для s/t/s-сигналов), и должно выполняться центральным ресурсом, а не агентом. Такое повышение напряжения может занимать от двух до трех тактов до полного восстановления SERR# [9].

Выводы прерываний

Прерывания на шине PCI произвольны и определяются как "чувствительные к уровню", т. е. устанавливаются по низкому логическому уровню, при этом для устройств используется выход с открытым коллектором. Переход сигналов INTx# в активное состояние и обратно асинхронно по отношению к сигналу CLK. Устройство выставляет прерывание по своей линии INTx#, когда требуется использование драйвера устройства, если устройство лишено возможности использовать сообщение, сигнализирующее о прерывании (MSI — message signaled interrupt). Если выставлен сигнал INTx#, он остается в активном состоянии, пока драйвер устройства не завершит незаконченный запрос. Когда запрос выполнен, устройство снимает сигнал INTx#.

PCI предусматривает одну линию прерываний для устройства с одной функцией, и до четырех линий прерывания — для многофункциональных[5] устройств или разъема. Для однофункционального устройства может использоваться только линия INTA#, в то время как три других линий прерывания не имеют никакого значения.

- **INTA#** (o/d). *Interrupt A* — используется для запроса прерывания.
- **INTB#** (o/d). *Interrupt B* — используется для запроса прерывания и имеет значение только для многофункционального устройства.
- **INTC#** (o/d). *Interrupt C* — используется для запроса прерывания и имеет значение только для многофункционального устройства.
- **INTD#** (o/d). *Interrupt D* — используется для запроса прерывания и имеет значение только для многофункционального устройства.

Любой функции на многофункциональном устройстве может быть назначена любая линия INTx#. Регистр вывода прерывания определяет, какая из линий INTx# используется для запроса прерывания. Название линий производится в алфавитном порядке. Если в устройстве используется единственная линия

[5] Устройство, в котором объединены несколько независимых функций, называется *многофункциональным устройством*. Каждая функция в таком устройстве обладает собственным конфигурационным пространством.

INTx#, то она называется INTA#; если реализуются две линии, то они называются INTA# и INTB#; и т. д. Все функции многофункционального устройства могут использовать одну и ту же линию INTx# для всех функций, либо каждая функция может иметь собственную линию (по максимальному количеству функций), или же любую комбинацию такого набора. Одна функция может генерировать запрос только на одной линии INTx#.

Способ объединения различных сигналов INTx# из разъема PCI, для их соединения с контроллером прерываний, произвольный. Они могут быть объединены по "ИЛИ", либо переключаться схемной логикой под управлением программы, либо как-то иначе, путем комбинации вышеперечисленных способов. Системный разработчик должен гарантировать, что каждый сигнал INTx# с каждого разъема подключается ко входу в контроллере прерывания. Все драйверы PCI-устройств должны обладать способностью к совместному использованию прерываний (цепочек прерываний) с любым другим логическим устройством, включая устройства в этом же самом многофункциональном модуле [9].

Пример соединений линий прерывания

Распределение прерываний на системной плате не принципиально. Приведенный далее пример показывает расчет при следующих условиях: система стандартна и контроллер прерывания имеет четыре открытых линии запросов прерывания. Так как большинство устройств однофункциональные и, следовательно, могут использовать только линию INTA#, то этот механизм распределяет прерывания равномерно среди входных выводов контроллера прерывания.

Линия INTA# устройства номер 0 связана с линией IRQW на системной плате. (Номер устройства не связан с расположением самого устройства на системной плате или в разъеме.) Линия INTA# устройства номер 1 связана с IRQX на системной плате. Линия INTA# устройства номер 2 связана с IRQY на системной плате. Линия INTA# устройства номер 3 связана с IRQZ на системной плате. Табл. 3.1 описывает связи каждой линии INTx# агента с линией прерывания системной платы. Чтобы определить, какая линия INTx# на системной плате соединена с линией INTx# данного устройства, может использоваться следующее уравнение:

MB = (D + I) MOD 4,

где MB — прерывание системной платы (IRQW = 0, IRQX = 1, IRQY = 2 и IRQZ = 3), D — номер устройства, I — номер прерывания (INTA# = 0, INTB# = 1, INTC# = 2 и INTD# = 3).

Таблица 3.1. *Соответствие линий прерываний*

Номер устройства на системной плате	Вывод прерывания на устройстве	Вывод прерывания на системной плате
0, 4, 8, 12, 16, 20, 24, 28	**INTA#**	**IRQW**
	INTB#	**IRQX**
	INTC#	**IRQY**
	INTD#	**IRQZ**
1, 5, 9, 13, 17, 21, 25, 29	**INTA#**	**IRQX**
	INTB#	**IRQY**
	INTC#	**IRQZ**
	INTD#	**IRQW**
2, 6, 10, 14, 18, 22, 26, 30	**INTA#**	**IRQY**
	INTB#	**IRQZ**
	INTC#	**IRQW**
	INTD#	**IRQX**
3, 7, 11, 15, 19, 23, 27, 31	**INTA#**	**IRQZ**
	INTB#	**IRQW**
	INTC#	**IRQX**
	INTD#	**IRQY**

Дополнительные выводы

- **PRSNT[1::2]#** (in). Вывод *Present* (присутствие) не предназначен для устройств и требуется для индикации наличия платы расширения в слоте. Сигнал PRSNT# указывает на наличие или отсутствие платы расширения в слоте. Если плата расширения присутствует, то с помощью данного сигнала передаются общие требования питания платы расширения. Данные сигналы обязательны для реализации платой расширения и необязательны для реализации на системной плате.
- **CLKRUN#** (in, o/d, s/t/s). *Clock running* — дополнительный сигнал, используемый в качестве входного для устройства и позволяющий определить статус линии CLK и выхода открытого коллектора, использованного устройством для запроса начала или ускорения CLK. CLKRUN# — подчиненный тристабильный сигнал. Используется центральным ресурсом для запроса на разрешение остановки или замедления CLK. Центральный ресурс отвечает за поддержание CLKRUN# в активном состоянии, пока вы-

ставлен сигнал CLK, и за снятие CLKRUN#, чтобы выполнить запрос на разрешение остановки или замедления CLK. Центральный ресурс должен обеспечивать необходимый уровень напряжения для CLKRUN#.

- **M66EN** (in). *66MHZ_ENABLE.* Сигнал указывает устройству, на какой частоте работает данный сегмент шины — 66 или 33 МГц.
- **PME#** (o/d). Сигнал *Power Management Event* является дополнительным сигналом, который может использоваться устройством для запроса на изменение состояния питания устройства или системы. Выставление и снятие сигнала PME# асинхронно по отношению к CLK. Этот сигнал имеет дополнительные электрические требования по сравнению со стандартными сигналами "открытый коллектор", что позволяет ему распространяться между отключенными и включенными устройствами. В общих чертах, этот сигнал распространяется между всеми разъемами PCI в системе, хотя при определенных условиях возможна отдельная передача буферизованной копии сигнала системной логике. Устройства должны быть инициализированы программным обеспечением перед выставлением этого сигнала. В случае выставления этого сигнала устройство должно удерживать сигнал в низком логическом уровне до тех пор, пока программное обеспечение явно не разрешит снять его.

  Производители должны обеспечивать необходимый уровень сигнала на этой линии, если предполагается ее использование. Производители, которые не используют эту линию, не должны выводить ее на разъемы, либо обеспечить соответствующий уровень сигнала.
- **3.3Vaux** (in). Дополнительный источник питания напряжением 3,3 В обеспечивает питание на платы расширения PCI. Предназначен для генерации событий управления питанием, когда основное питание на плате снято программно. Системные платы или платы расширения, которые не поддерживают управление питанием PCI, должны рассматривать вывод 3.3Vaux как зарезервированный [9].

Реализация линий PRSNT#, CLKRUN#

На плате расширения как минимум один из двух выводов PRSNT[1::2]# должен быть заземлен. Данное требование обеспечивает возможность указания системной плате, что плата расширения физически присутствует в разъеме. Уровень сигналов PRSNT1# и PRSNT2# сообщает системной плате требования питания платы расширения. В простейшем случае данные выводы платы расширения PRSNT1# и/или PRSNT2# могут быть соединены с землей, чтобы сообщать плате соответствующие требования питания. При отсутствии платы расширения системная плата должна обеспечивать на этих линиях соответствующий уровень напряжения.

Сигнал CLKRUN# используется в мобильном окружении PCI и не предназначен для использования в разъеме. Подробности протокола CLKRUN# и другие соглашения описаны в спецификации *PCI Mobile Design Guid*e [9, 12].

Выводы расширения шины до 64 бит

Выводы расширения до 64 бит не обязательны для реализации. При использовании такого расширения необходимо задействовать все выводы, описанные в данном разделе.

- **AD[63::32]** (t/s). *Address Data* (адрес и данные), как и младшие разряды, мультиплексированы на одних и тех же выводах и обеспечивают 32 дополнительных разряда. Во время фазы адреса (когда используются команды двойного адресного цикла и когда активен сигнал REQ64#) передаются старшие 32 бита 64-разрядного адреса; в противном случае эти биты зарезервированы[6], но при этом стабильны и не определены. Во время фазы данных, когда 64-битная транзакция подтверждена активностью сигналов REQ64# и ACK64#, передаются дополнительные 32 бита данных.
- **C/BE [7::4]#** (t/s). *Bus Command* и *Byte Enables* (команды шины и разрешение обращения к байтам) мультиплексированы на одних и тех же выводах. Во время фазы адреса (когда используются команды двойного адресного цикла и когда активен сигнал REQ64#) передается фактическая команда шины по линиям C/BE[7::4]#; в противном случае эти биты зарезервированы и не определены. В течение фазы данных по линиям C/BE[7::4]# передается полубайт, который показывает, какой байт данных содержит значимые данные, при условии, что активны оба сигнала REQ64# и ACK64#. Сигнал C/BE[4]# применяется к байту 4, а сигнал C/BE[7]# — к байту 7.
- **REQ64#** (s/t/s). *Request 64-bit Transfer.* Управляется мастером шины, показывает, что тот собирается передать данные, используя для пересылки 64 бита. Сигнал REQ64# имеет такие же временные параметры, что и FRAME#. Кроме того, этот сигнал также имеет определенное значение в конце сброса.
- **ACK64#** (s/t/s). *Acknowledge 64-bit Transfer.* Управляется устройством, которое успешно дешифрировало данный адрес в качестве агента текущего доступа, показывает, что агент собирается передать данные, используя при этом 64 бита. Сигнал ACK# имеет такие же временные параметры, как и DEVSEL#.

---

[6] Это означает резервирование данных разрядов комитетом PCI SIG для будущего использования. Зарезервированные биты не должны быть использованы каким-либо устройством.

- **PAR64** (t/s). *Parity Upper DWORD.* Бит контроля по четности[7] линий AD[63::32] и C/BE[7::4]. Сигнал PAR64 действует в течение одного такта после фазы начального адреса, пока активен REQ64. Сигнал PAR64 стабилен и корректен для 64-битных фаз данных в течение одного такта, когда активен сигнал IRDY# при транзакции записи, либо когда активен TRDY# при транзакции чтения. (PAR64 имеет те же временные параметры, что и AD[63::32], но с задержкой на один такт.) Мастер управляет сигналом PAR64 во время фазы адреса и фазы записи данных; агент управляет PAR64 во время фаз чтения данных [9].

Выводы JTAG

Выводы JTAG (Joint Test Automation Group, объединенная рабочая группа по автоматизации тестирования) предназначены для контроля и отладки устройств. Стандарт IEEE 1149.1 (определяющий интерфейс JTAG), *порт для тестирования и архитектура периферийного сканирования ("Test Access Port and Boundary Scan Architecture")*, включен в качестве необязательного интерфейса для PCI-устройств. Стандарт IEEE 1149.1 определяет правила и ограничения для проектирования ИС (интегральных схем) в соответствии с этим стандартом. Включение в состав устройства *порта для тестирования (TAP — Test Access Port)* позволяет использовать периферийное сканирование для проверки устройства и платы, на которой установлено данное устройство. TAP состоит из четырех выводов (в общем случае — из пяти), которые применяются для организации последовательного интерфейса с контроллером TAP внутри PCI-устройства.

- **TCK** (in). *Test Clock* используется для синхронизации ввода в устройство собранной информации и данных и их вывода во время работы с TAP.
- **TDI** (in). *Test Data Input* используется для последовательного ввода в устройство тестирующих данных и команд при работе с TAP.
- **TDO** (out). *Test Output* используется для последовательного вывода тестирующих данных и команд из устройства при работе с TAP.
- **TMS** (out). *Test Mode Select* используется для управления в устройстве состоянием контроллера TAP.
- **TRST#** (in). *Test Reset* обеспечивает асинхронную инициализацию контроллера TAP. Этот сигнал по стандарту IEEE 1149.1 необязателен.

Данные выводы TAP должны работать при тех же параметрах электрической среды (5 или 3,3 В), что и буферы ввода-вывода PCI интерфейса устройств.

---

[7] Количество (сумма) "1"-ц на линиях AD[63::32], C/BE[7::4]# и PAR64 должно быть равно четному числу.

Кроме того, управление выводом TDO необязательно должно быть таким же, как это делается для стандартных выводов шины PCI. Способ управления TDO должен быть указан в техническом паспорте устройства [9].

Производитель системы ответственен за проектирование и функционирование в системе последовательных цепочек стандарта IEEE 1149.1 (так называемые "кольца"). Дополнительные сигналы в "многоточечном" режиме на шине PCI не используются. Обычно "кольцо" по стандарту IEEE 1149.1 создается путем соединения вывода TDO одного устройства с выводом TDO другого, чтобы получить последовательную цепочку устройств. В этом случае микросхемы получают одни и те же сигналы TCK, TMS и необязательные сигналы TRST#. Все кольца по стандарту IEEE 1149.1 соединены либо с тестирующим разъемом материнской платы с целью тестирования, либо с ИС резидентного контроллера соответствующего этому стандарту.

Спецификация PCI поддерживает платы расширения с разъемом, который предусматривает сигналы периферийного сканирования. Устройства на плате расширения могут соединяться в цепочку на материнской плате. Методы соединения и использования системы колец по стандарту IEEE 1149.1 с платами расширения включают:

- использование кольца по стандарту IEEE 1149.1 на плате расширения только во время тестирования этой платы расширения на производстве. В этом случае, кольцо по стандарту IEEE 1149.1 на материнской плате не должно контактировать с сигналами, соответствующими этому стандарту для плат расширения. Материнская плата должна проходить самотестирование непосредственно в ходе производства;
- создание для каждого разъема платы расширения отдельного кольца на материнской плате по стандарту IEEE 1149.1. Например, если на плате есть два разъема расширения, то на материнской плате должны оставаться свободные кольца по этому стандарту;
- инициализацию ИС, которая допускает иерархичную многоточечную адресацию по стандарту IEEE 1149.1. Это позволит обрабатывать множество колец по этому стандарту и разрешит выполнение многоточечных операций.

Платы расширения, не поддерживающие стандарт интерфейса IEEE 1149.1, должны осуществлять переход от вывода TDI платы к выводу TDO [9].

Выводы интерфейса SMBus

Выводы интерфейса SMBus применяются совместно. Если реализованы дополнительные возможности управления, то требуются обе линии SMBCLK и SMBDAT.

- **SMBCLK** (o/d). Дополнительный интерфейсный сигнал синхронизации интерфейса SMBus. Этот вывод зарезервирован для дополнительной поддержки SMBCLK.
- **SMBDAT** (o/d). Дополнительный интерфейсный сигнал данных интерфейса SMBus. Этот вывод зарезервирован для дополнительной поддержки SMBDAT [9, 16].

Внеполосные сигналы

Описанные линии и сигналы на них позволяют организовать все необходимые механизмы взаимодействия компонентов и устройств на шине, включая такой специфический вид доступа, как доступ нескольких мастеров.

Тем не менее существует возможность введения собственных линий и сигналов, называемых *внеполосными*, или *пользовательскими*. В качестве таких сигналов могут использоваться любые сигналы, не описанные в спецификации PCI, но которые соединяют два или более PCI-агентов и имеют значение только для них. Данные сигналы могут использоваться для одного или большего количества устройств с целью объединения их специфических состояний и обеспечения максимальной эффективности использования в системе шины PCI.

Эти сигналы не предусматриваются в разъеме PCI. Следовательно, они должны быть ограниченны средой системной платы. Кроме того, данные сигналы могут нарушать специфицированный протокол для определенных сигналов PCI, либо приводить к таким нарушениям протокола [9].

Функции центрального ресурса

Спецификация шины PCI вводит термин *центральный ресурс,* который используется для описания поддерживаемых функций шины, обычно обеспечиваемых в системе PCI мостом, либо стандартным интерфейсом. К этим функциям относятся следующие:

- центральный арбитраж (REQ# — входной и GNT# — выходной);
- приведение требуемых сигналов в активное состояние;
- вычитающее дешифрирование. Только один агент на шине PCI может использовать вычитающее дешифрирование и, как правило, таким устройством является мост к стандартной шине расширения;
- преобразование процессорной транзакции в транзакцию конфигурации;
- генерирование индивидуальных сигналов IDSEL для каждого устройства в целях конфигурирования системы;
- управление сигналом REQ64# во время инициализации.

Пример контроллера PCI

В качестве примера использования пользовательских (вспомогательных) сигналов рассмотрим контроллер моста PCI/памяти фирмы Atmel [5].

Контроллер TSPC106 предназначен для организации интерфейса между процессором PowerPC 60x, кэшем второго уровня (или до 4-х дополнительных процессоров PowerPC 60x), шиной PCI и основной памятью. Поддержка PCI позволяет разрабатывать системы на основе готовых решений, разработанных для PCI-периферии. Контроллер TSPC106 использует 3,3 В КМОП-технологию и обеспечивает полную интерфейсную совместимость с ТТЛ-устройствами.

TSPC106 интегрирует возможности тестирования и отладки через интерфейс периферийного сканирования JTAG. Структурная схема контроллера представлена на рис. 3.3.

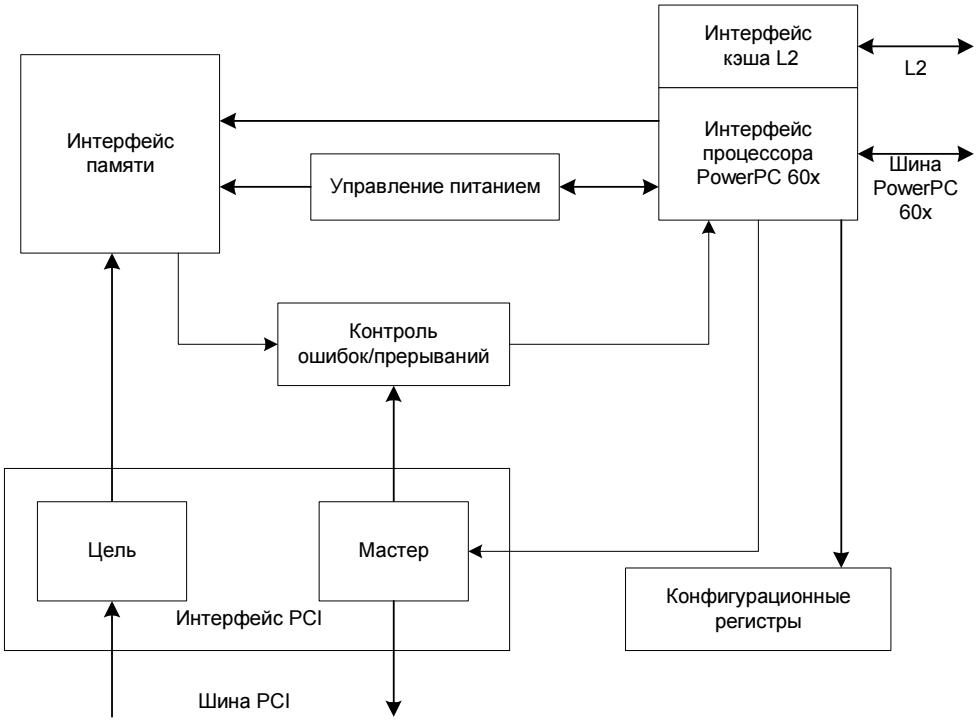


Рис. 3.3. Структурная схема контроллера TSPC106

Микросхема TSPC106 представляет CHRP-совместимый (CHRP — Common Hardware Reference Platform — прежнее обозначение платформы PowerPC) мост микропроцессора PowerPC между семейством микропроцессоров PowerPC

и шиной PCI. CHRP объединяет в себе множество спецификаций, которые определяют объединенную архитектуру персональных компьютеров. Стандарт комбинирует преимущества платформы Power Macintosh и стандартного окружения PC.

Контроллер TSPC106 поддерживает программируемый интерфейс к разнообразным микропроцессорам PowerPC, управляя выбором скорости шины. Разрядность шины адреса процессора PowerPC 60x составляет 32 бита; шины данных — 64 бита. Процессорный интерфейс контроллера TSPC106 использует подмножество протоколов шины PowerPC 60x, обеспечивает побитную и поблочную передачи данных. Адресная шина и шина данных разделены для поддержки раздельных транзакций. TSPC106 обеспечивает поддержку для следующих конфигураций 60x-процессоров и кэш-памяти второго уровня (L2):

- ❒ до 4 процессоров PowerPC 60x без кэш-памяти L2;
- ❒ единственный 60x-процессор и кэш второго уровня, при этом используется внутренний контроллер кэша L2 в TSPC106;
- ❒ до 4 процессоров PowerPC 60x и внешнеуправляемый кэш второго уровня (например, интегрированный кэш L2 Motorola MPC2604GA).

Интерфейс памяти управляется процессором и способен к поддержке разнообразных конфигураций, использующих типы памяти DRAM, EDO, или SDRAM и ROM, или FlashROM.

PCI-интерфейс контроллера TSPC106 совместим со спецификацией PCI версии 2.1. Интерфейс PCI соединяет процессорную и шину памяти с шиной PCI, с которой связаны компоненты ввода-вывода. Шина PCI использует 32-разрядные мультиплексированные шины данных/адреса плюс различные сигналы управления и контроля ошибок. Интерфейс PCI контроллера TSPC106 функционирует как мастер и как цель (как ведущее устройство и как ведомое). Как мастер, TSPC106 поддерживает операции чтения и записи при обращении в область памяти PCI, область ввода-вывода и конфигурационное пространство PCI. Контроллер TSPC106 также поддерживает специальный цикл PCI и команды подтверждения прерываний. Как цель, он поддерживает операции чтения и записи в системную память. Контроллер TSPC106 обеспечивает аппаратную поддержку для 4-х уровней пониженного потребления питания. Интерфейс TSPC106 является полностью статическим, позволяя внутренним логическим состояниям быть сохраненным во время действия режима пониженного потребления питания [5].

Сигналы на контроллере TSPC106 (рис. 3.4) группированы следующим образом:

- ❒ сигналы интерфейса процессора PowerPC 60x;
- ❒ сигналы интерфейса кэша L2 /многопроцессорной поддержки;

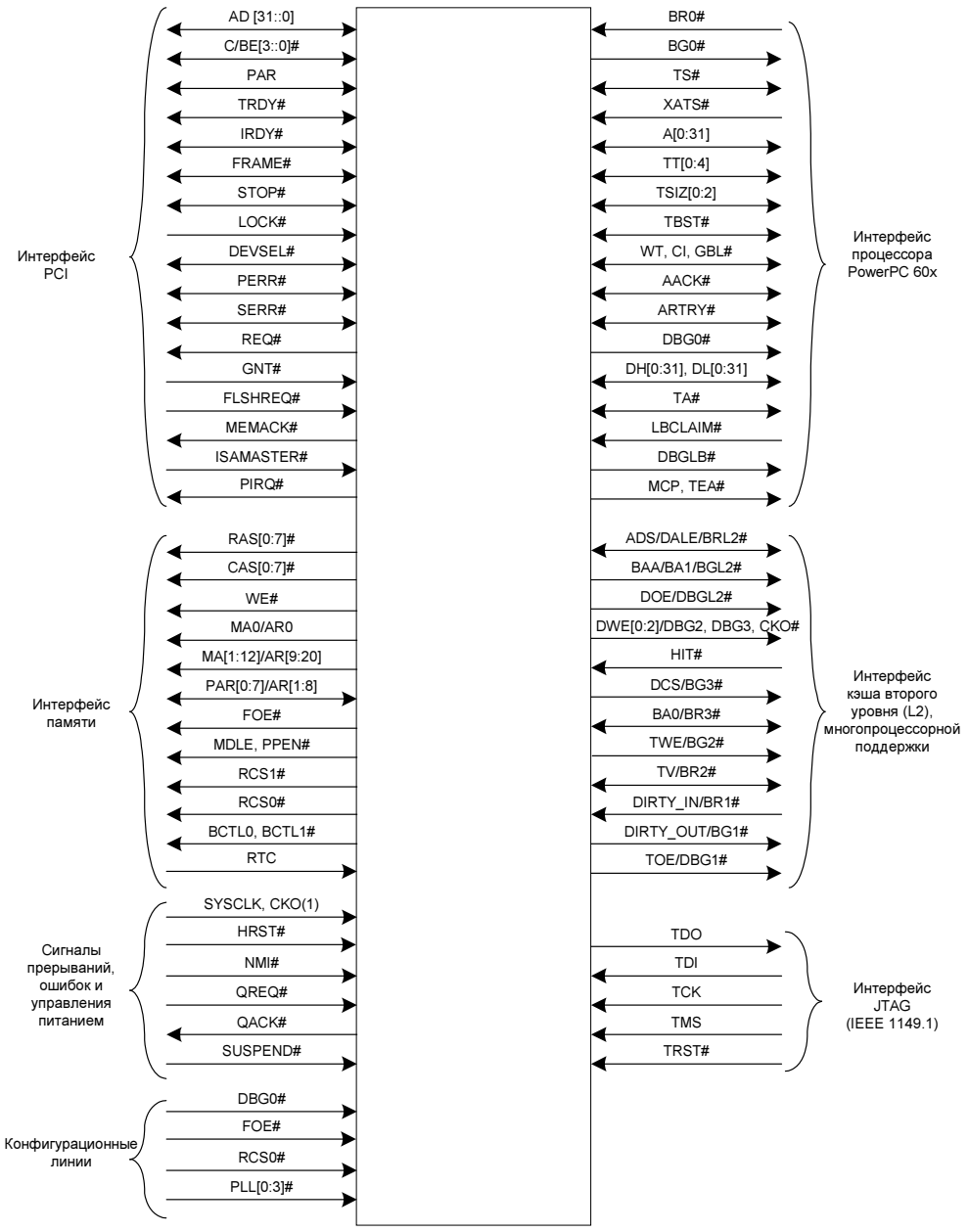


Рис. 3.4. Сигналы контроллера TSPC106

- сигналы интерфейса памяти;
- сигналы интерфейса PCI;
- сигналы прерывания, синхронизации и управления питанием;
- сигналы интерфейса IEEE 1149.1;
- сигналы конфигурации.

Особый интерес представляет то, что интерфейс контроллера TSPC106 использует 4 пользовательских сигнала PCI: FLSHREQ, ISA_MASTER, MEMACK и PIRQ.

- **FLSHREQ** (*Flush request*, один вывод, входной) показывает, что устройство должно взаимодействовать с контроллером TSPC106; сбрасывает все его текущие операции.
- **ISA_MASTER** (*ISA master*, один вывод, входной) показывает, что мастер шины ISA обращается к системной памяти.
- **MEMACK** (*Memory acknowledge*, один вывод, выходной) показывает, что контроллер TSPC106 сбросил все свои текущие операции и заблокировал все передачи процессора PowerPC 60x, кроме операции фонового копирования. Контроллер TSPC106 выставляет сигнал MEMACK в ответ на сигнал FLSHREQ после того, как выполнено сбрасывание.
- **PIRQ** (*Modified memory interrupt reques*, один вывод, выходной) в режиме эмуляции, показывает, что запись в память на шине PCI не была произведена [5].

Команды

Взаимодействие между различными PCI-устройствами осуществляется посредством управляющих воздействий в виде команд шины PCI. Команда представляет собой 4-значное число в двоичной системе счисления. Каждому числу сопоставляется выполнение той или иной операции. Команды передаются по линиям C/BE[3::0]# с помощью соответствующих сигналов. Во время передачи адреса по линиям AD по линиям C/BE[3::0]# передается команда, а во время передачи данных по линиям AD, по линиям C/BE[3::0]# передается разрешение обращения к байтам. Основное назначение команд состоит в том, чтобы мастер мог указать агенту тип запрашиваемой транзакции. При формировании и исполнении команд устанавливаются некоторые правила и ограничения [9].

Назначение и классификация команд

Всего на шине PCI определено 16 команд (из 4 линий C/BE[3::0]# возможно составить только 16 комбинаций). Команды формируются мастером и пред-

назначены для указания агенту типа запрашиваемой транзакции. Команда передается по линиям C/BE[3::0]# одновременно с передачей адреса. В зависимости от типа адресного пространства, команды условно делятся на:

- ❒ команды работы с памятью;
- ❒ команды работы с пространством конфигурации;
- ❒ специализированные команды.

Команды работы с памятью и осуществляют операции чтения/записи в память. Команды, предназначенные для работы с пространством конфигурации, осуществляют операции чтения/записи в него. Кроме того, существуют команды, предназначенные для повышения производительности [9, 18].

Коды команд на шине PCI (табл. 3.2), их типы и краткое описание каждой из них приведены далее. Коды команд показаны в таком виде, в каком они присутствуют на шине ("1" показывает высокий логический уровень напряжения, "0" — низкий). Сигнал разрешения обращения к байтам активен при низком логическом уровне напряжения ("0").

Таблица 3.2. *Команды шины*

C/BE[3::0]#	Тип операции
0000	*Interrupt Acknowledge*
0001	*Special Cycle*
0010	*I/O Read*
0011	*I/O Write*
0100	Зарезервировано
0101	Зарезервировано
0110	*Memory Read*
0111	*Memory Write*
1000	Зарезервировано
1001	Зарезервировано
1010	*Configuration Read*
1011	*Configuration Write*
1100	*Memory Read Multiple*
1101	*Dual Address Cycle*
1110	*Memory Read Line*
1111	*Memory Write and Invalidate*

- Команда *Interrupt Acknowledge* (подтверждение прерывания) представляет собой неявное обращение к системному контроллеру прерываний. Под неявным обращением в данном случае понимается считывание вектора прерывания из контроллера прерываний. Во время фазы адреса адресные линии не имеют логического значения, сигнал разрешения обращения к байтам показывает длину возвращаемого вектора.
- Команда *Special Cycle* (специальный цикл) предназначена для осуществления простого механизма широковещательной передачи сообщений по шине PCI. Суть этого механизма состоит в следующем. Команда *Special Cycle* не содержит никакого явного адреса и передается всем агентам. Каждый агент, получивший сообщение, должен определить, применимо ли оно к нему. Во время фазы данных по линиям AD[31::00] передается тип сообщения и сами данные. Сообщение закодировано на младших 16 линиях, а именно AD[15::00]. Необязательное поле данных закодировано на старших 16 линиях AD[31::16] и используется не во всех сообщениях. Уникальность этой команды по сравнению с другими заключается в том, что ни один из агентов не выставляет сигнал DEVSEL#, и транзакция заканчивается с аварийным завершением (так называемый "Master-Abort", правила и порядок его применения рассмотрены в *главе 4*). Данный механизм используется в качестве альтернативного физическим сигналам, когда необходимо организовать связь по вспомогательным линиям.
- Команда *I/O Read* (чтение пространства ввода-вывода) предназначена для чтения данных, поступающих от устройства. В данном случае взаимодействие с устройством производится через пространство ввода-вывода. По адресным линиям AD[31::00] поступает адрес байта, который будет считан. Должны дешифрироваться все 32 разряда. Сигнал разрешения обращения к байтам указывает размер передачи, при этом он должен соответствовать адресу байта (т. е. этот сигнал указывает, какие из передаваемых по линиям AD[31::00] байтов имеют значение).
- Команда *I/O Write* (запись в пространстве ввода-вывода) предназначена для передачи данных устройству, отображенному в адресном пространстве ввода-вывода. Так же как и для предыдущей команды, должны дешифрироваться все 32 бита, а сигнал разрешения обращения к байтам указывает размер передачи и должен соответствовать адресу байта.
- *Зарезервированные* коды команд предназначены для будущего использования. PCI-устройства не должны использовать эти коды, а также не должны на них отвечать или реагировать. Если по линии C/BE# передаются какие-либо из зарезервированных кодов, то доступ должен быть завершен способом Master-Abort (аварийное завершение, инициированное мастером).

- Команда *Memory Read* (чтение из памяти) предназначена для чтения данных от агента, отображенного в пространстве адресов памяти. Цель (целевое устройство) может выполнять упреждающее чтение (чтение, при котором читается не только конкретный адрес, но и смежные с ним) для этой команды, при условии, что при таком чтении не будет никаких побочных эффектов. Кроме того, для данной PCI-транзакции целевое устройство должно проверить синхронизацию данных, хранящихся во временных буферах. Эти буферы должны быть очищены перед любыми событиями синхронизации (например, обновление регистра состояния ввода-вывода или флага памяти) при использовании данной команды.
- Команда *Memory Write* (запись в память) используется для передачи данных агенту, отображенному в пространстве адресов памяти. Когда целевое устройство выставляет сигнал готовности к обмену (сигнал TRDY#), то оно принимает ответственность за порядок (включая упорядочивание) подчиненных данных. Это может обеспечиваться как выполнением данной команды полностью синхронным способом, так и проверкой любого программного буфера до его очищения перед каким-то событием синхронизации (например, при обновлении регистра состояния ввода-вывода или флага памяти). В данном случае подразумевается, что мастер может инициировать событие синхронизации сразу после использования этой команды.
- Команда *Configuration Read* (чтение конфигурации) используется для чтения пространства конфигурации, уникального для каждого агента. Агент является выбранным, когда активен его сигнал IDSEL, и логическое значение линий AD[1::0] равно "00". Во время фазы адреса конфигурационной транзакции, по линиям AD[7::2] передается адрес одного из 64 регистров размером в двойное слово (при этом сигнал разрешения обращения к байтам указывает байт или несколько байтов внутри каждого DWORD, т. е. двойного слова) в конфигурационном пространстве устройства. Линии AD[31::11] игнорируются. Линии AD[10::08] предназначены для адресации устройства, если основное устройство является многофункциональным.
- Команда *Configuration Write* (запись конфигурации) используется для передачи данных в пространство конфигурации устройства. Порядок адресации для транзакций записи конфигурации такой же, как и для транзакций чтения конфигурации.
- Команда *Memory Read Multiple* (множественное чтение памяти) является семантически идентичной команде *Memory Read*, за исключением того, что в ней дополнительно указывается возможность выбора мастером более чем одной строки кэш-памяти перед выполнением. Контроллер памяти должен продолжать конвейерную обработку памяти, пока активен сигнал

FRAME#. Эта команда предназначена для работы по передаче больших последовательностей данных. За счет ее использования возможно повысить эффективность системы памяти (и запрашивающего мастера) при последовательном чтении дополнительной строки кэш-памяти, когда программно установленный буфер доступен для временного хранения.

- Команда *Dual Address Cycle* (или иначе *DAC* — двойной цикл адреса) используется для передачи 64-разрядного адреса в устройства, поддерживающие 64-битную адресацию. Используется в случае, когда требуемый адрес находится за пределами нижнего пространства размером 4 Гбайта. Устройства, которые поддерживают только 32-битную адресацию, должны обработать эту команду, как зарезервированную, и не отвечать на текущую транзакцию.
- Команда *Memory Read Line* (чтение полной строки памяти) является семантически идентичной команде *Memory Read*. В этой команде дополнительно указывается, что мастер собирается выбирать полную строку кэш-памяти. Эта команда предназначена для работы с большими последовательностями передаваемых данных. За счет ее применения возможно повысить производительность системы (и запрашивающего мастера), при чтении строки кэш-памяти до конца, в том случае, если время реакции на запрос составляет менее одного цикла чтения памяти. Как и в случае с командой *Memory Read*, буферы выборки должны быть очищены до того, как будут инициированы события синхронизации.
- Команда *Memory Write and Invalidate* (запись в память с аннулированием) семантически идентична команде *Memory Write*, за исключением того, что она дополнительно гарантирует передачу как минимум одной полной строки кэш-памяти; например, это требуется, когда мастеру необходимо записать все байты внутри адресованной строки кэш-памяти за одну транзакцию PCI. Мастер может позволить транзакции "перейти" границу строки кэш-памяти только в случае, если он предполагает передачу и всей последующей строки. Для выполнения этой команды требуется наличие у мастера регистра конфигурации, в котором был бы указан размер строки кэш-памяти. Это позволит повысить производительность памяти путем экономии строки в кэш-памяти с обратной записью, без осуществления фактического цикла обратной записи, и, таким образом, сократить время доступа [9, 18].

Правила и ограничения использования команд

Формирование, передача и исполнение команд предполагает наличие некоторых условий и ограничений.

Для всех устройств, размещающихся на шине PCI, определено следующее требование. На команды с кодом "1010" и "1011" (чтение и запись простран-

ства конфигурации) все устройства должны отвечать как целевые (ведомые устройства). Данное требование не распространяется на главный мост шины. К остальным командам не предъявляется таких требований и, по сути, они являются необязательными.

Мастер и цель (целевое устройство) могут выполнять любые необязательные команды, но для целей существует отдельное правило. Если же цель реализует основные команды работы с памятью (т. е. в интерфейсе цели предусмотрено формирование таких команд и обработка результатов их выполнения), то необходимо поддерживать все команды данного типа, включая *Memory Write and Invalidate*, *Memory Read Line*, *Read Multiple*. Три перечисленные команды предназначены для повышения производительности. При усеченной реализации (когда эти команды не используются) они должны быть совмещены с основными командами работы с памятью. Например, цель может не реализовывать команду *Memory Read Line*; тем не менее, она должна принять запрос (если адрес дешифрирован как адрес для доступа к памяти) и обработать его как команду чтения из памяти *Memory Read*. Аналогично, цель может не реализовывать команду *Memory Write and Invalidate*, но она обязана принять запрос (если адрес дешифрирован как адрес для доступа к памяти) и обработать его как команду *Memory Write*.

Для передачи блоков данных в системную память и обратно мастеру рекомендуется поддерживать команды *Memory Write and Invalidate*, *Memory Read Line*, *Memory Read Multiple*. Однако если по каким-либо причинам мастер не способен использовать данные команды, то допускается использовать только команды *Memory Read* или *Memory Write*. Для мастеров, использующих команды чтения данных из памяти, существует правило: для всех команд будет работать доступ любого размера. Это означает, что запрос любой длины будет обработан корректно. Их основное применение показано далее.

Команда записи в память с аннулированием — *Write and Invalidate* — единственная команда, для исполнения которой требуется регистр длины строки кэш-памяти. Настоятельно рекомендуется, чтобы команды чтения памяти использовали этот регистр.

Прежде чем вводить следующее правило, необходимо сказать, что понимается под чтением с упреждением и выборкой по запросу. Последовательность действий алгоритма *выборки по запросу* следующая:

1. Мастер, обращаясь через мост к каким-либо данным, указывает их адрес.
2. Мост производит чтение указанных данных и передает их мастеру.

Упреждающее чтение или *чтение с упреждением* представляет собой модификацию алгоритма выборки, т. е., кроме запрошенных данных, мост производит чтение области памяти, окружающей эти данные (при страничной организации памяти — так называемый *кластер*). Такой алгоритм призван

уменьшить накладные расходы, связанные с большим количеством исключительных ситуаций, возникающих при работе с большими объемами данных или кода. Кроме того, оптимизируется и работа с устройством (памятью, диском), к которому обращается мастер, поскольку появляется возможность загрузки нескольких частей данных за одно обращение.

При упреждающем чтении данных мастер может и не использовать остальные считанные данные (которые он не запрашивал, но мог бы запросить). Такие неиспользованные данные являются скрытыми по отношению к мастеру. Мост, который производит чтение с упреждением, несет ответственность за любые скрытые данные, не использованные мастером. Рекомендации команд памяти изменяются в зависимости от характеристик местоположения памяти и количества читаемых данных. Местоположения памяти разделяются на:

- ❒ доступные для чтения с упреждением;
- ❒ не доступные.

Память, доступная для чтения с упреждением, имеет следующие характеристики:

- ❒ отсутствуют побочные эффекты операции чтения. Операция чтения не может разрушить данные или какую-либо статусную информацию. Например, *буфер FIFO* (First In, First Out), в котором при обращении к следующим данным во время чтения, замещаются устаревшие данные, не может быть упреждающим. Точно так же любой адрес, при обращении по которому очищается бит состояния, не будет доступен для упреждающего чтения;
- ❒ во время операции чтения устройство обязано возвращать все байты независимо от состояния сигнала разрешения обращения к байтам (четыре или восемь байт в зависимости от размера передачи данных);
- ❒ мостам разрешено производить объединение несвязанных байтов в этом диапазоне.

Память другого типа, т. е. не отвечающая перечисленным требованиям, рассматривается, как недоступная для упреждающего чтения.

В табл. 3.3 приводятся рекомендуемые использования для команд чтения.

С точки зрения адресата (цели), данные команды чтения должны рассматриваться так же, даже при условии, что они не обращаются к первому двойному слову строки кэш-памяти. Например, цель, к которой производится обращение к двойному слову 1 (вместо двойного слова 0), должна будет произвести упреждающее чтение до конца текущей строки кэш-памяти. Если регистр длины строки кэш-памяти не реализован, то мастер должен установить размер строки 16 или 32 байта и использовать команды чтения, рекомендованные в табл. 3.3 (это обеспечивает линейный порядок в блоке) [9].

Таблица 3.3. *Использование команд чтения*

Команда	Условия предпочтительного использования
Memory Read	Чтение в диапазоне адресов с возникновением побочных эффектов (память, не доступная для упреждающего чтения); чтение отдельного двойного слова (DWORD)
Memory Read Line	Чтение данных, превышающих размер двойного слова (DWORD) до следующей границы строки кэш-памяти в адресном пространстве, доступном для упреждающего чтения
Memory Read Multiple	Чтение блока данных, который выходит за границу строки кэш-памяти в адресном пространстве, доступном для упреждающего чтения

Правила использования команд чтения

Различные команды чтения будут оказывать разное влияние на работу системы, потому что главные мосты и мосты PCI-to-PCI должны рассматривать команды по-разному. Когда используется команда *Memory Read*, мост получит только те данные, которые были запрошены мастером, т. к. может существовать побочный эффект (данные могут быть разрушены при чтении из памяти, не доступной для упреждающего чтения). Мост не имеет возможности производить чтение с упреждением, потому как неизвестно, какие байты потребуются для следующей фазы данных. Эта информация не будет доступна, пока не закончится текущая фаза данных. Однако, для команд *Memory Read Line* и *Memory Read Multiple*, мастер гарантирует, что диапазон адреса, к которому он обращается, доступен для упреждающего чтения. При этом условии мосту разрешено производить чтение с упреждением и получить больше данных, чем фактически запросил мастер. Такой способ обращения к памяти позволяет увеличить производительность системы.

Как пример, предположим, что мастеру необходимо прочитать три двойных слова от цели, находящейся с другой стороны моста PCI-to-PCI. Если мастер использовал команду *Memory Read*, то мост не может читать второе двойное слово от цели, потому что он не располагает следующим сигналом разрешения обращения к байтам. Транзакция завершится после одной передачи данных. Но если бы мастер использовал команду *Memory Read Line*, мост смог бы читать данные от цели блоками до конца строки кэш-памяти, за счет чего данные поступили бы к мастеру быстрее.

Команда *Memory Read Multiple* также позволяет мостам производить чтение данных с упреждением. Для этого мостом должен быть реализован регистр

размера строки кэш-памяти, чтобы гарантировать правильное использование команд чтения. Регистр размера строки кэш-памяти должен быть реализован при использовании дополнительного упорядочивания блока в режиме Cacheline Wrap. Использование правильной команды чтения позволяет получить оптимальную производительность. Если же задействованы не все команды чтения, то выбираются наиболее производительные для данной системы.

Например, если мастер при чтении в основном считывает полную строку кэш-памяти, и только изредка считывает больше одной строки кэш-памяти, то для этого устройства было бы разумно использовать только команду *Memory Read Line* для обоих типов доступа.

Как определялось ранее, мост, который производит упреждающее чтение, отвечает за любые скрытые данные, не использованные мастером. Самый простой метод правильной обработки скрытых данных состоит в том, чтобы просто пометить их как незначимые в конце текущей транзакции [9].

Возможные последствия упреждающего чтения

Предположим, что в памяти выделено два буфера в смежных участках оперативной памяти. Центральный процессор готовит сообщение мастеру шины в первом буфере и затем сообщает ему, чтобы он забрал сообщение. Когда мастер читает его сообщение, мост между мастером и основной памятью производит чтение с упреждением последующих адресов, включая и часть второго буфера. Некоторое время спустя центральный процессор готовит второе сообщение, используя второй буфер в оперативной памяти, и сообщает мастеру, чтобы он получил его. Если мост не пометил скрытые данные, прочитанные перед этим, как незначимые, то, когда мастер попытается прочитать второй буфер, мост может вернуть данные, считанные во время предыдущего чтения из второго буфера. Таким образом, второе сообщение мастер не получит.

Такая же ситуация возникает, если устройству необходимо опросить участок памяти, находящийся с другой стороны моста. Устройство никогда не получило бы новые данные из этого участка, если бы мост не очищал буфер после каждой операции чтения. Таким образом, устройство получало бы одни и те же данные дважды, не получая истинных данных вообще [9].

Правила и порядок адресации

Всего в интерфейсе PCI определено три физических адресных пространства:

- пространство адресов памяти;
- пространство ввода-вывода;
- конфигурационное пространство (пространство конфигурации).

Пространства адресов памяти и ввода-вывода объединены. Конфигурационное адресное пространство должно присутствовать в каждом устройстве, и предназначено для поддержки аппаратной конфигурации PCI. Работа с этими пространствами описана далее.

Целевые PCI-устройства (исключая главные мосты шины) должны содержать *регистр базового адреса*. Значение, содержащееся в этом регистре, используется при запросе диапазона адресов, которые могут быть использованы для обеспечения доступа к внутренним регистрам или функциям устройства. Конфигурационное программное обеспечение (ПО) использует регистр базового адреса для определения размера диапазона адресов, требуемого устройству в данном адресном пространстве, и затем назначает (если возможно) место в адресном пространстве, через которое будет происходить взаимодействие с устройством.

При возникновении транзакции на интерфейсе каждое потенциальное целевое устройство сравнивает адрес, передаваемый в транзакции, со значением, хранящемся в своем регистре базового адреса. Таким способом целевое устройство (цель) определяет, является ли она целью текущей транзакции. Если адрес совпал, то устройство выставляет сигнал DEVSEL#, и к нему будет осуществлен доступ.

Все из указанных пространств различаются по функциональному предназначению, следовательно, различается и схема дешифрации адреса для каждого из них. Дешифрация адреса целью в каждом адресном пространстве рассмотрена в следующих разделах. Присутствуют два вида дешифрации:

- прямая (иначе позитивная);
- вычитающая (иначе субтрактивная) [9].

Дешифрация пространства ввода-вывода

Под *пространством ввода-вывода* понимается отдельное адресное пространство, отличное от адресного пространства физической памяти. Стандартно адресное пространство ввода-вывода состоит из 64К индивидуально адресуемых 8-битных портов; любые два последовательно расположенные 8-битных порта могут рассматриваться как один 16-битный, а любые четыре — как один 32-битный порт. Физически порту ввода-вывода соответствует аппаратный регистр.

Алгоритм дешифрации для пространства ввода-вывода выглядит в общем случае так. Дешифрируются все 32 адресные линии, что обеспечивает адресацию байта в диапазоне 0x00000000 до 0xFFFFFFFF. Мастер, который инициирует транзакцию, должен гарантировать, что линии AD[1::0] указывают младший значащий правильный байт для транзакции.

Сигнал разрешения обращения к байтам BE# показывает размер передаваемых данных, а также незначимые байты в пределах двойного слова (DWORD), и должен соответствовать значению AD[1::0]. По сути, этот сигнал показывает, какой из 4-х байтов двойного слова является начальным для транзакции. Табл. 3.4 показывает допустимые комбинации для линий AD[1::0] и соответствующее значение сигнала BE# для начальной фазы данных.

Таблица 3.4. *Комбинации сигнала разрешения обращения к байтам BE#*

AD[1::0]	Начальные (младшие) байты	Допустимые комбинации BE#[3:0]
00	Байт 0	xxx0 или 1111
01	Байт 1	xx01 или 1111
10	Байт 2	x011 или 1111
11	Байт 3	0111 или 1111

Значение "0" указывает выбор байта, т. е., для байта разрешения величина xx01 означает, что начальным байтом транзакции является байт 1, байт 0 не используется, байты 2 и 3 не имеют значения.

Примечание

Если на линиях BE#[3::0] присутствует значение "1111", т. е. все линии находятся в неактивном состоянии, AD[1::0] может принимать любое значение. В этом случае сигнал разрешения обращения к байтам BE# не указывает номер байта в двойном слове.

Функция устройства может ограничить количество поддерживаемых типов доступа. Например, на уровне драйвера устройство может определить, чтобы доступ к функции осуществлялся с использованием операции размером в байт, слово или двойное слово, все остальные доступы устройство может закончить с аварийным завершением. Комбинации AD[1::0] и BE#[3::0] являются специфическими для каждого устройства и могут использоваться, чтобы определить, какие доступы нарушают поддерживаемую адресацию [9].

Дешифрация адресного пространства памяти

Шина PCI обладает 32 адресными линиями и позволяет адресовать до 4 Гбайт физической памяти (2^{32}–1). Адрес передается по линиям AD[31::02], чем достигается его выравнивание на границу двойного слова. Суть *выравнивания адреса* на двоичном уровне состоит в следующем. Два младших байта не участвуют в дешифрации и их значение с точки зрения дешифратора всегда равно 0. Адрес изменяется, начиная с третьего байта. Например, адрес первого

двойного слова будет 0...000b, адрес второго двойного слова будет 0...100b, третьего 0...1000b. В шестнадцатеричной системе счисления данные значения эквивалентны 0h, 4h, 8h и, следовательно, адрес всегда будет выровнен на размер двойного слова.

Значения линий AD[1::0] не являются частью дешифрируемого адреса. Тем не менее AD[1::0] указывает порядок, в котором должны передаваться данные. Этот порядок запрашивается мастером. Табл. 3.5 показывает режим в блоке данных, запрашиваемый мастером при использовании команд для работы с памятью, линии AD[1::0] указывают порядок.

Таблица 3.5. *Режимы в блоке данных*

AD1	AD0	Режим в блоке
0	0	Линейное приращение
0	1	Зарезервировано (разрыв связи после первой фазы данных)[8]
1	0	Режим переключения строки кэш-памяти (Cacheline Wrap)
1	1	Зарезервировано (разрыв связи после первой фазы данных)

Под *линейным приращением* или режимом в блоке при AD[1::0]="00" понимается прямое увеличение адреса, на 4 байта для 32-разрядной адресации и на 8 байтов для 64-разрядной. Всем целям требуется проверять линии AD[1::0] во время команды транзакции при обращении к памяти, а также обеспечивать требуемую последовательность передаваемого блока. Если цель (целевое устройство) не поддерживает режим в блоке, запрошенный мастером, цель должна выполнить одну фазу данных и затем завершить запрос с типом прекращения транзакции "Disconnect". Это гарантирует, что транзакция будет выполнена (хотя и медленно, поскольку каждый запрос выполнится как одна транзакция фазы данных). Цель должна использовать завершение "Disconnect with Data" в течение первой фазы данных или "Disconnect without Data" для второй фазы данных. При таком способе завершения данные передаются только в одной фазе данных. Неспособность поддержки специфического режима в блоке не является основанием для завершения транзакции целью с типом прекращения транзакции "Retry".

Если цель поддерживает блоки на шине (burst-режим), то она должна поддерживать линейное упорядочение блока. Поддержка обратной строки кэша

[8] Эта величина зарезервирована и не может использоваться. В ранних версиях спецификации PCI данная величина имела значение и мастера, которые генерировали ее, а также некоторые цели, могли обеспечить продолжение транзакции.

является необязательной. Транзакции, в которых используется команда *Memory Write and Invalidate*, могут использовать только линейный монопольный режим приращения.

Режим переключения строки кэш-памяти или, иначе, "Cacheline Wrap" — это режим, при котором обращение может начинаться с любого места внутри строки кэш-памяти и до конца, затем происходит обращение в ее начало и до места, с которого оно начиналось. После этого можно переключаться на другую строку кэш-памяти. Доступ происходит за счет увеличения адреса на двойное слово (для 64-разрядной транзакции на два двойных слова) и выполняется до тех пор, пока не будет достигнут конец строки кэш-памяти, затем происходит обращение в начало этой же строки. Это продолжается, пока не будет передана остальная часть строки. Длина строки кэш-памяти определяется значением, хранящимся в регистре размера строки кэш-линии в пространстве конфигурации, и инициализируется конфигурационным программным обеспечением. Например, для доступа, в котором размер строки кэш-памяти составляет 16 байт (четыре двойных слова) и транзакция адресована к двойному слову по смещению 08h, последовательность для 32-разрядной транзакции была бы следующей:

1. Первая фаза данных адресована к двойному слову по смещению 08h.
2. Вторая фаза данных адресована к двойному слову по смещению 0Ch, которое является концом текущей строки кэш-памяти.
3. Третья фаза данных адресована к двойному слову по смещению 00h, которое является началом адресуемой строки кэш-памяти.
4. Последняя фаза данных адресована к двойному слову по смещению 04h, которая завершает доступ ко всей строке кэш-памяти.

Если после завершения доступа к данной строке кэш-памяти блок продолжается, то обращение выполняется по тому же смещению двойного слова, с которого начинался доступ, но при этом добавляется адрес следующей строки кэш-памяти. Продолжение блока предыдущего примера было бы следующим:

1. Пятая фаза данных адресована к двойному слову по смещению18h.
2. Шестая фаза данных адресована к двойному слову по смещению 1Ch, которая завершает вторую строку кэш-памяти.
3. Седьмая фаза данных адресована к двойному слову по смещению 10h, которая начинает вторую строку кэш-памяти.
4. Последняя фаза данных адресована к двойному слову 14h, которая завершает доступ ко второй строке кэш-памяти.

Если цель не содержит регистр размера строки кэш-памяти, то она должна указать состояние "Disconnect" одновременно или после завершения первой

фазы данных. Если мастер инициирует транзакцию с одним типом порядка в блоке, он не может его изменить до конца текущей транзакции, т. к. информация о порядке в блоке была передана по линиям AD[1::0] во время фазы адреса.

Устройство может ограничить размер доступа к пространству памяти. Например, устройство посредством драйвера может поддерживать только доступы размером в байт, слово, или двойное слово, и завершать доступы другого размера с типом прекращения транзакции "Target-Abort" [9].

Дешифрация конфигурационного пространства

Конфигурационное пространство физически представляет собой 256 регистров размером 32 байта, которое должно присутствовать в каждом PCI-устройстве, за исключением главных мостов шин. Основное назначение конфигурационного пространства заключается в предоставлении возможности программной инициализации и конфигурации устройства. Реализация конфигурационного адресного пространства в главных мостах не обязательна. Пространство конфигурации назначается для каждой функции устройства.

В следующих разделах описаны команды конфигурации (тип 0 и тип 1), программная генерация команд конфигурации, программная генерация специальных циклов, выбор пространства конфигурации устройства и порядок генерирования сигнала IDSEL.

Организация иерархических шин. Конфигурационные транзакции

Из-за электрических вопросов нагрузки число устройств, которые могут поддерживаться в одном сегменте шины, ограничено. Увеличение количества PCI-устройств в системе возможно осуществить за счет использования нескольких взаимосвязанных сегментов шины. Для подключения дополнительных сегментов шины необходимо соединяющее звено — мост PCI-to-PCI. Основная функция моста заключается в организации взаимодействия между устройствами, находящимися в разных сегментах шины. Следовательно, мост должен распознавать транзакции, происходящие внутри сегмента, и транзакции, адресованные в другой сегмент шины. Для поддержки иерархических шин PCI введены два типа конфигурационных транзакций — "тип 0" и "тип 1". Их форматы проиллюстрированы на рис. 3.5, который показывает интерпретацию линий AD во время фазы адреса конфигурационной транзакции.

Тип 1 и тип 0 конфигурационных транзакций отличаются значениями AD[1::0]. Конфигурационная транзакция типа 0 (AD[1::0] = "00") использует-

ся для выбора устройства в том же самом сегменте, где будет происходить транзакция. Конфигурационная транзакция типа 1 (AD[1::0] = "01") используется, чтобы передать конфигурационный запрос в другой сегмент шины.

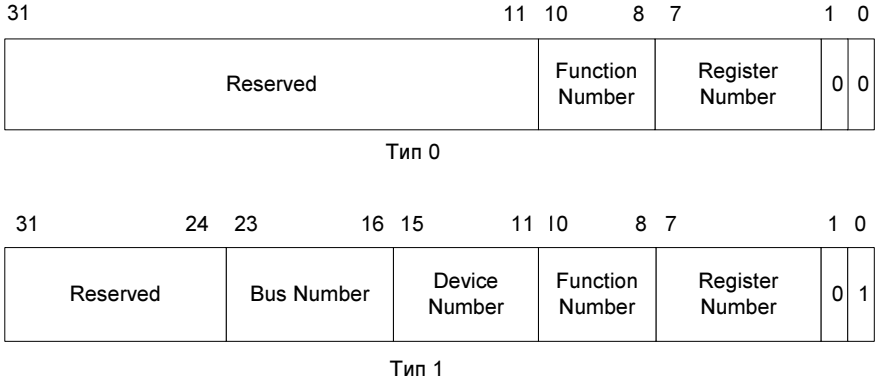


Рис. 3.5. Форматы фаз адреса конфигурационных транзакций

Значение, записываемое в регистр CONFIG_ADDRESS (0xCF8), интерпретируется мостом в соответствии с рис. 3.5. При этом двоичное представление записанной величины условно разделяется на поля с определенным смещением (например, для доступа к регистру № 3C устройства № 0x1E на шине 0, функция 0 в регистр 0xCF8 необходимо записать значение 0x8000F03C). Поля **Register Number** и **Function Number** имеют одно и то же значение для обоих типов транзакций, поля **Device Number** и **Bus Number** используются только в транзакции типа 1. Зарезервированные поля цель должна игнорировать. Значение полей следующее:

- **Register Number** — адрес регистра в пространстве конфигурации предполагаемой цели передается по линиям AD[7::2];
- **Function Number** — номер одной из 8-ми возможных функций в многофункциональном устройстве передается по линиям AD[10::08];
- **Device Number** — номер одного из 32 устройств на данной шине;
- **Bus Number** — номер одной из 256 шин в системе.

Мосты (как главные, так и PCI-to-PCI), которые производят генерацию конфигурационной транзакции типа 0, используют поле **Device Number** для выбора устройства. Выбор устройства для конфигурационной транзакции осуществляется по сигналу IDSEL.

Конфигурационная транзакция типа 0 не распространяется за локальной шиной PCI и должна предъявляться локальным устройствам. Если объект конфигурационной транзакции находится на другой шине, должна использовать-

ся конфигурационная транзакция типа 1. Все цели внутри данного сегмента шины, кроме мостов PCI-to-PCI, игнорируют конфигурационную транзакцию типа 1.

Транзакции типа 0 и типа 1 обрабатываются следующим образом. Мост PCI-to-PCI дешифрирует поле **Bus Number**, чтобы определить, где находится требуемая шина конфигурационной транзакции. Если значение поля **Bus Number** не указывает на шину, находящуюся с другой стороны моста, транзакция игнорируется. В этом случае транзакция происходит без изменений. Если транзакция адресована к шине, находящейся с другой стороны моста, мост при передаче преобразует транзакцию в конфигурационную транзакцию типа 0. При этом мост изменяет значение AD[1::0] на "00" и передает AD[10::02] без изменений. Поле **Device Number** дешифрируется для выбора одного из 32 устройств на шине. Мост выставляет сигнал IDSEL, соответствующего устройства и инициирует конфигурационную транзакцию типа 0 [9].

Программная генерация транзакций конфигурации

В различных случаях устройствам необходимо обращаться в пространство конфигурации без непосредственного использования механизма[9] аппаратного обеспечения. Для этого разработан специальный механизм, который позволяет программному обеспечению генерировать конфигурационные транзакции на шине PCI. Этот механизм реализуется за счет использования пространства ввода-вывода (порты обычно размещаются в главном мосте). Для PC/AT-совместимых систем генерирование транзакций конфигурации определено и описывается в данном разделе.

Суть механизма состоит в том, что устройство обращается в пространство ввода-вывода, используя операнды размером в двойное слово. Комбинация определенного адреса в пространстве ввода-вывода и обращение к нему размером в двойное слово служит указанием для моста, что выполняется механизм программной генерации конфигурационной транзакции. Фактически обращение происходит в регистры главного моста. Драйвер устройства должен использовать *интерфейс API* (Application Programming Interface), предоставляемый операционной системой, чтобы обратиться в пространство конфигурации устройства. При таком подходе обращение происходит без непосредственного использования механизма аппаратного обеспечения. Для других системных архитектур данный механизм не определен.

Рассмотрим операции, выполняемые для реализации данного механизма. Для генерации транзакций конфигурации используются два диапазона простран-

[9] В версиях 2.0 и 2.1 спецификации PCI было определено два механизма. Только один из них (конфигурационный механизм 1) был разрешен для новых разработок и второй (механизм 2) был включен для рекомендации.

ства ввода-вывода (для PC/AT-совместимых систем) размером в двойное слово каждый. Первый диапазон по адресу 0xCF8h указывает на регистр, предназначенный для чтения/записи, который называется CONFIG_ADDRESS. Второй адрес 0xCFCh указывает на регистр, предназначенный также для чтения/записи, называемый CONFIG_DATA. Регистр CONFIG_ADDRESS является 32-разрядным, его формат приведен на рис. 3.6. Здесь Бит 31 — флаг разрешения, определяющий, что доступ к регистру CONFIG_DATA должен быть преобразован в транзакцию конфигурации на шине PCI. Биты от 30 до 24 зарезервированы, доступны только для чтения, и должны возвращать значение "0" при чтении. Биты от 23 до 16 определяют шину PCI. Биты от 15 до 11 определяют устройство на шине, биты от 10 до 8 указывают определенную функцию в устройстве (если устройство поддерживает несколько функций). Биты от 7 до 2 указывают двойное слово в пространстве конфигурации устройства, т. е. позволяют адресовать 64 двойных слов (256 байт). Биты 1 и 0 доступны только для чтения и при обращении должны возвратить значение "0".

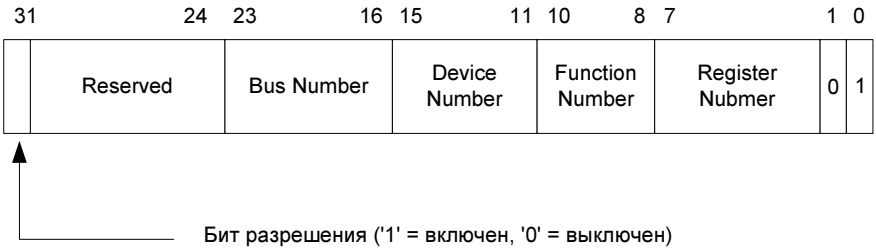


Рис. 3.6. Формат регистра CONFIG_ADDRESS

Как только через главный мост проходит запись двойного слова в пространство ввода-вывода по адресу 0xCF8, мост должен зафиксировать эти данные в его регистре CONFIG_ADDRESS. При чтении двойного слова пространства ввода-вывода по адресу 0xCF8 мост должен возвратить данные из регистра CONFIG_ADDRESS. Другие типы доступов в этот адрес (размер которых не равен двойному слову) не оказывают влияния на CONFIG_ADDRESS и выполняются как обычные транзакции ввода-вывода на шине PCI. Устройства ввода-вывода, совместно осуществляющие доступ по этому адресу, но использующие регистр размером байт или слово, не будут оказывать влияния на данный механизм, т. к. их транзакции пройдут через мост без изменений.

Когда через мост выполняется доступ в пространство ввода-вывода, адрес которого находится внутри двойного слова, начинающегося с адреса 0xCFC (CONFIG_DATA), он проверяет бит разрешения и номер шины в регистре CONFIG_ADDRESS. Если бит разрешения установлен, и номер шины соответствует номеру шины моста или любому номеру шины, находящейся с дру-

гой стороны моста, должно быть выполнено преобразование цикла конфигурации. Так как существует два типа конфигурационных транзакций, то имеют место два типа преобразования.

- *Преобразования типа 0* — выполняются, если адресуемое устройство находится на шине PCI, соединенной с главным мостом.
- *Преобразования типа 1* — выполняются, если шина, на которой находится устройство, располагается с другой стороны моста.

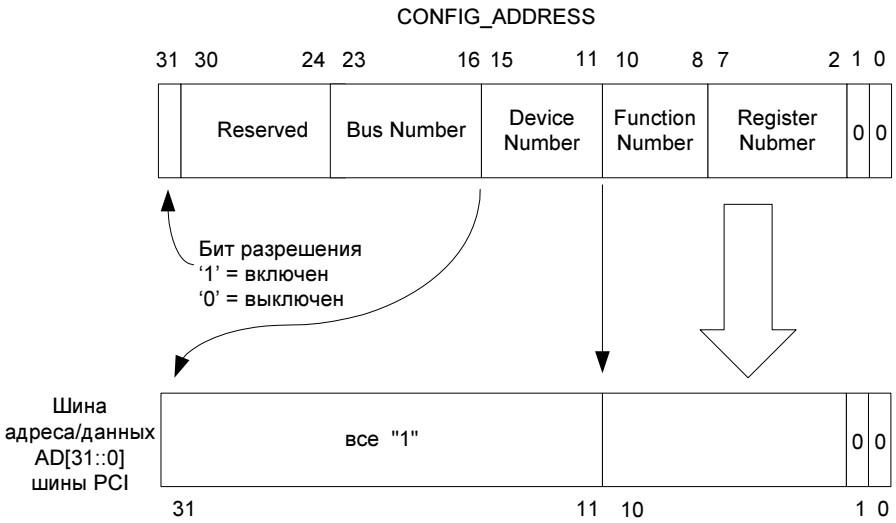


Рис. 3.7. Преобразование типа "0"

Главный мост выполняет преобразование типа 0 следующим образом (рис. 3.7). Мост дешифрирует значение поля **Device Number**, чтобы выставить сигнал на соответствующую IDSEL линию[10]. Таким образом, из линий AD[31::11] только на одной будет присутствовать значение "1", определяющее конкретное устройство. Биты 10—8 из регистра CONFIG_ADDRESS копируются на линии AD[10::8]. На линии AD[7::2] также будут скопированы значения из регистра CONFIG_ADDRESS. При этом значение AD[1::0]="00". На рис. 3.7 показано преобразование из регистра CONFIG_ADDRESS на адресные линии шины PCI [9].

[10] Если поле **Device Number** указывает на отсутствующую линию IDSEL, то мост должен выполнить доступ процессора как обычно, пропуская данные для записи и возвращая все единицы при чтении. Мост также может выполнить конфигурационный доступ типа 0 без выставления сигнала по линии IDSEL. Это приведет к завершению транзакции типа "Master-Abort", в результате которого пропускаются данные для записи и возвращаются все единицы при чтении.

Чтобы выполнить преобразование типа 1, главный мост копирует содержание регистра CONFIG_ADDRESS (исключая биты 31 и 0) на адресные линии PCI во время адресной фазы конфигурационной транзакции. При этом значение AD[1::0] = "01". Для обоих типов преобразований байт разрешения для фаз данных должен быть скопирован непосредственно с шины процессора.

Поддержка одноранговых шин

Для главных мостов, которые не поддерживают одноранговые шины, выполнение преобразований конфигурационных доступов в транзакции осуществляется достаточно просто. Если значение поля **Bus Number** в регистре CONFIG_ADDRESS равно 0, используется преобразование конфигурации типа 0. Во всех других случаях (значение поля **Bus Number** отлично от нуля) выполняется алгоритм конфигурационного преобразования типа 1.

Распределение функций между мостами, поддерживающими одноранговые шины, производится следующим образом. Один одноранговый мост обычно назначается, чтобы всегда распознавать доступ в регистр CONFIG_ ADDRESS. Другой одноранговый мост должен следить за данными, записанными в этот регистр. Обращение в регистр CONFIG_DATA осуществляется "рукопожатием" с мостом, выполняющим преобразование транзакции конфигурации. В главных мостах, которые поддерживают одноранговые шины, должны присутствовать два регистра в пространстве конфигурации. Один регистр, BUS NUMBER, определяет номер шины PCI, находящейся непосредственно с другой стороны моста. Другой регистр, SUBORDINATE BUS NUMBER, определяет номер последней иерархической шины PCI в системе. В пространстве конфигурации моста PCI-to-PCI должен присутствовать дополнительный регистр PRIMARY BUS NUMBER, определяющий номер его основной шины. Конфигурационное программное обеспечение системы должно производить инициализацию этих регистров соответствующими значениями. Главный мост определяет тип преобразования конфигурации ("1" или "0") на основании значения номера шины в регистре CONFIG_ADDRESS. Если значение поля **Bus Number** в регистре CONFIG_ADDRESS соответствует значению в регистре BUS NUMBER, то используется конфигурационная транзакция типа 0. Если номер шины в CONFIG_ADDRESS превышает номер шины в регистре BUS NUMBER и меньше или равен номеру шины в регистре SUBORDINATE BUS NUMBER, то используется конфигурационная транзакция типа 1. Если же значение поля **Bus Number** в регистре CONFIG_ADDRESS меньше значения в регистре BUS NUMBER или больше чем значение в регистре SUBORDINATE BUS NUMBER, то транзакция конфигурации адресована в шину, которая отсутствует или находится позади другого главного моста и игнорируется [9].

Программная генерация специального цикла

Программная генерация транзакций конфигурации позволяет прозрачно осуществлять доступы в конфигурационное пространство различных устройств. Для обеспечения большей гибкости определен механизм программной генерации специального цикла, при этом доступы должны осуществляться в те же регистры, что и для конфигурационных транзакций.

Рассмотрим порядок выполнения данной операции. В регистр CONFIG_ ADDRESS производится запись следующего вида. Значение поля **Bus Number** соответствует номеру шины моста, значение полей **Device Number** и **Function Number** равно "11111", значение поля **Register Number** равно 0. Такая запись служит предписанием для моста, указывающая, что во время следующей записи в регистр CONFIG_DATA он должен генерировать транзакцию специального цикла. Когда в регистр CONFIG_DATA произведена запись, мост генерирует транзакцию, которая использует команду специального цикла. Код команды передается (до передачи команды *Configuration Write*) по линиям C/BE[3::0]# в течение фазы адреса. После того, как в регистр CONFIG_ADDRESS будет произведена запись такого типа, чтение из регистра CONFIG_DATA будет иметь неопределенные результаты. Допускается обработка данной ситуации как нормальной конфигурационной операции — мост генерирует конфигурационную транзакцию типа 0, которая завершается с "Master-Abort". При этом по линиям данных возвращаются все "1".

Если поле **Bus Number** регистра CONFIG_ADDRESS не соответствует номеру шины, на которой находится мост, то мост передает запись в CONFIG_DATA как конфигурационную транзакцию типа 1. Так же мост обрабатывает запросы, когда не совпадают номера шин [9].

Выбор пространства конфигурации устройства

Для обращения в пространство конфигурации какого-либо устройства необходимо знать его адрес, т. е. номер шины, номер устройства и номер функции. Операция "выбор шины" осуществляется стандартно. Операция "выбор устройства" для конфигурационного доступа носит специфический характер. Для выбора устройства при осуществлении доступа в конфигурационное пространство был введен специальный сигнал — IDSEL, который функционирует как классический сигнал "chip select" (выбор кристалла). Данный сигнал является входным для всех устройств и должен присутствовать в каждом из них. Исключением являются главные мосты шины, которым разрешено выполнить выбор внутренне. Сигнал IDSEL каждого из устройств соединяется с одной из 21 старших адресных линий главного моста, т. е. с одной из AD[31::11]. Данные линии не используются во время конфигурационной транзакции. Запрещается осуществлять внутреннее соединение между лини-

ей IDSEL и линиями AD в устройстве в целях экономии выводов. Единственным исключением из этого правила является главный мост, т. к. он определяет, как отображаются линии IDSEL. Порядок соединения линий AD и линий IDSEL не определен, т. е. оставлен на усмотрение разработчиков системы. Также не установлена связь поля **Device Number** регистра CONFIG_ ADDRESS для PC/AT-совместимых систем и генерированием сигнала IDSEL, в результате чего BIOS должна сканировать все 32 номера устройства для гарантированного обнаружения всех существующих устройств. Генерация сигнала IDSEL с другой стороны моста PCI-to-PCI определена в спецификации PCI-to-PCI.

Аппаратное обеспечение, которое преобразовывает номер устройства в IDSEL, должно контролировать формирование сигнала IDSEL только на одной линии. Конфигурационные транзакции, которые не были востребованы устройством, завершаются с "Master-Abort". Мастер, который инициировал эту транзакцию, устанавливает полученный бит **Master-Abort** в регистре состояния.

Таким образом, данный алгоритм предполагает внешнее выполнение выбора устройства при доступе в конфигурационное адресное пространство. После выполнения операции выбора устройство должно принять либо отвергнуть транзакцию, запрашиваемую мастером. При этом устройства разделяются на два типа и различаются значениями в заголовке пространства конфигурации. Первый тип — *однофункциональные устройства* — определен для обратной совместимости. Устройства этого типа для определения поведения используют только выводы IDSEL и AD[1:: 0]. Однофункциональное устройство выставляет сигнал DEVSEL#, чтобы требовать конфигурационной транзакции при выполнении следующих условий:

- ❒ дешифрирована команда конфигурации;
- ❒ сигнал на линии IDSEL устройства находится в активном состоянии;
- ❒ значение AD[1::0] равно "00" (конфигурационная транзакция типа 0) во время фазы адреса.

В противном случае, устройство игнорирует текущую транзакцию. Однофункциональное устройство может отвечать при обращении к другим номерам функций как та же самая функция. Устройство может дешифрировать поле **Function Number** адресных линий и отвечать только на функцию № 0, а при обращении к другим номерам функций не отвечать (что приведет к завершению транзакции способом "Master-Abort"). Допускаются оба варианта реализации.

Второй тип устройств — *многофункциональные устройства*. Для определения поведения устройство данного типа дешифрирует поле **Function Number** (линии AD[10::08]), чтобы выбрать одну из восьми возможных функций в

устройстве. Многофункциональные устройства обязаны выполнять полную дешифрацию линий AD[10::08] и отвечать на цикл конфигурации только в том случае, если для выбранной функции реализованы регистры пространства конфигурации. Они не должны отвечать на запросы с номерами несуществующих функций (что приведет к завершению транзакции способом "Master-Abort"). В таких устройствах всегда должна присутствовать функция 0. Реализация других функций необязательна и может быть назначена в любом порядке (т. е. устройство с двумя функциями должно ответить на запрос к функции 0, но вторая функции может иметь любой из возможных номеров функций от 1 до 7). Многофункционально устройство выставляет сигнал DEVSEL#, чтобы требовать конфигурационной транзакции при выполнении следующих условий:

- ❒ дешифрирована команда конфигурации;
- ❒ сигнал на линии IDSEL устройства находится в активном состоянии;
- ❒ значение AD[1::0] равно "00";
- ❒ значение AD[10::08] указывает на существующую функцию.

Во всех других случаях транзакция игнорируется. Например, в устройстве реализованы функции 0 и 4 (функции от 1 до 3 и от 5 до 7 — нет). Для транзакции конфигурации, в которой выставлен сигнал IDSEL, значение AD[1:: 0] = "00", и значение AD[10::08] = "000" или "100", устройство выставило бы сигнал DEVSEL#. Линии AD[31::11] игнорируются многофункциональным устройством во время доступа к его регистрам пространства конфигурации.

Порядок, в котором программное обеспечение нумерует устройства, постоянно находящиеся в сегменте шины, не определен. Обычно конфигурационное программное обеспечение начинает с Device Number 0 и далее по возрастанию или наоборот, начинает с Device Number 31 и далее по убыванию. Если обнаружено однофункциональное устройство (т. е. бит 7 в регистре HEADER TYPE функции 0 равен 0), то для остальных функций поле **Device Number** не проверяется. Если обнаружено многофункциональное устройство (т. е. бит 7 в регистре HEADER TYPE функции 0 равен 1), то будут проверены все поля **Function Number**.

Как только функция была выбрана, она использует линии AD[7::2] для адресации двойного слова и сигнал разрешения обращения к байтам, чтобы определить, к каким байтам обращаться внутри адресованного двойного слова. Функция не должна ограничивать размер доступа, который поддерживается в пространстве конфигурации. Команды конфигурации, как и другие команды, позволяют передавать данные, используя любую комбинацию байтов в блоке (включая байт, слово, двойное слово или неупорядоченную последовательность байтов), а также несколько фаз данных в блоке. Цель (целевое устрой-

ство) должна обработать любую комбинацию байта разрешения. Цель не обязана обрабатывать конфигурационную транзакцию, которая состоит из нескольких фаз данных. Если транзакция конфигурации состоит из более чем одной фазы данных, то цель может завершить запрос с типом завершения транзакции "Disconnect". Завершение с типом "Target-Abort" при этом не допускается, т. к. данное состояние не является состоянием ошибки. Для блочной конфигурационной транзакции единственный допустимый способ адресации — линейное упорядочивание в блоке. В этом случае по линиям AD[1::0] передается тип конфигурационной транзакции, а не способ адресации блока подобно доступу в адресное пространство памяти. Предполагаемый адрес каждой последующей фазы данных на одно двойное слово больше, чем адрес предшествующей. Например, транзакция начинается с передачи по линиям AD[7::2] двоичных значений "0000 00xx", при этом последовательность блока была бы следующей: "0000 01xx", "0000 10xx", "0000 11xx", "0001 00xx" (где xx указывают тип конфигурационной транзакции 00 или 01). Конфигурационная транзакция типа 1, которая преобразована в транзакцию, использующую команду специального цикла, может быть только одиночной [9].

Рекомендации соединения линии IDSEL

Порядок назначения номеров устройствам определяется разработчиком. Тем не менее, если к системе не предъявляются специфические требования, то может использоваться следующий пример. Линия IDSEL устройства 0 соединена с линией AD[16], IDSEL устройства 1 соединена с линией AD[17], и т. д., линия IDSEL устройства 15 соединена с линией AD[31]. Для устройств 16—31 главный мост должен выполнить конфигурационную транзакцию, не выставляя сигнала ни на одной из линий AD[31::16]. Доступ завершится со способом завершения транзакции "Master-Abort".

Для конфигурационного доступа может быть выбрано 21 устройство путем соединения различных адресных линий с каждым устройством. Соединение одной из 21 старших адресных линий AD[11::31] с линией IDSEL создает дополнительную нагрузку на адресной линии. Ее возможно снизить за счет резистивной связи IDSEL и соответствующей адресной линии. Но такая связь приводит к очень медленной скорости изменения сигнала на линии IDSEL, что приводит к его нахождению в недопустимом логическом состоянии большинством времени, как показано на рис. 3.8 с метками "XXXX". Но, поскольку данный сигнал используется только в адресной фазе конфигурационной транзакции типа 0, адресная шина может быть "предуправляемой" несколько тактов перед выставлением сигнала FRAME#[11], за счет чего гаранти-

[11] Число тактов, во время которых должно происходить предуправление адресной шины, определяется из константы RC на IDSEL.

руется устойчивое состояние сигнала IDSEL. Предупреждение адресной шины эквивалентно продвижению IDSEL и, по сути, представляет собой задержку для "медленного" сигнала. В случае использования резистивной связи мост, который генерирует транзакцию конфигурации, должен использовать продвижение IDSEL или гарантировать, что период такта достаточно большой, чтобы позволить сигналу IDSEL перейти в устойчивое состояние до начала конфигурационной транзакции. Для всех других циклов сигнал IDSEL не определен и может находиться в недетерминированном уровне во время адресной фазы [9].

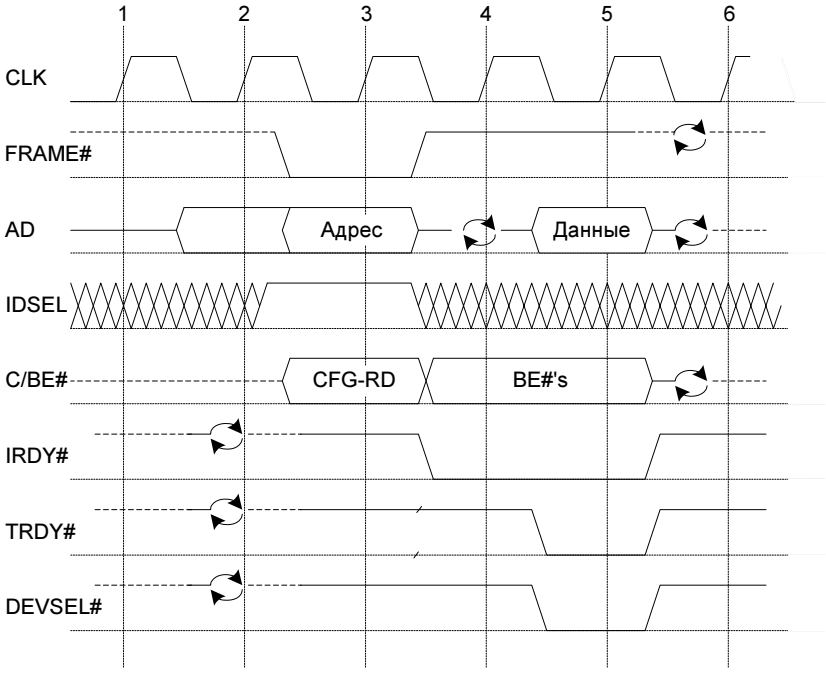


Рис. 3.8. Конфигурационная транзакция чтения

Дешифрация пространства ввода-вывода для Legacy-устройств

В современных системах до сих пор присутствуют некоторые унаследованные интерфейсы и технологии, такие как, например, шина ISA (теперь LPC). Следовательно, необходимо обеспечить обратную совместимость с такими устройствами. В пространстве ввода-вывода для этих целей выделены некоторые диапазоны адресов, доступ в которые необходимо описать. В качестве

примера приведем часть таблицы из *спецификации PC System Design Guide 2001* (табл. 3.6).

Таблица 3.6. *Пример ISA адресов ввода-вывода*

Адрес ввода-вывода	Системная функция по умолчанию
0000–000F	Slave DMA
0010–0018	System
0001F	System
0020–0021	Master 8259
0170–0177	Secondary IDE controller
01F0–01F7	Primary IDE controller
03BC–03BE	LPT 3 (XT parallel port 1)
03C0–03DF	EGA/VGA
0CF8–0CFB	PCI ports

Устройство, которое поддерживает унаследованную функцию PC (IDE, VGA, и т. д.), может требовать обращения к этим адресам, т. к. они связаны с определенной функцией. Для этого должен быть установлен бит разрешения пространства ввода-вывода.

При таком обращении в пространство ввода-вывода устройство не использует регистры базового адреса (за исключением моста шины расширения). То есть, требует *Legacy-адресов* (т. е. традиционных унаследованных адресов) пространства ввода-вывода каждый раз, когда установлен его бит разрешения пространства ввода-вывода. Legacy-адреса не требуют использования регистра базового адреса и назначаются инициализационным программным обеспечением. Устройство идентифицирует свою функцию как Legacy, посредством указания соответствующего значения в поле "Class Code" пространства конфигурации.

При доступе в пространство ввода-вывода адрес может выходить за пределы Legacy-диапазона ввода-вывода. В этом случае устройство обязано использовать линии AD[1::0], чтобы указать недопустимые байты до окончания дешифрации. После этого выставляется сигнал DEVSEL#. Если Legacy-функция адресует транзакцию ввода-вывода, но не все байты в пределах двойного слова попадают в допустимый диапазон, то она обязана закончить транзакцию с типом "Target-Abort". Мосту шины расширения предоставлено исключение из этого требования, когда выполняется вычитающая дешифрация. Допускается предположение, что все адресованные байты в пределах

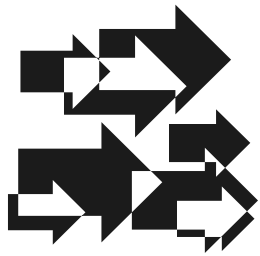
двойного слова находятся на шине расширения. Это означает, что мост не обязан проверять линии AD[1::0] и сигнал разрешения обращения к байтам до окончания передачи запроса на шину расширения [9].

Рекомендации распределения адресного пространства устройств

Число различных устройств постоянно увеличивается, примером чего может служить расширение классов и типов поддерживаемых PCI-устройств. Поэтому настоятельно рекомендуется, чтобы внутренние регистры устройства отображались в пространство памяти, а не в пространство ввода-вывода. Данное отображение реализуется за счет регистра базового адреса. В настоящее время пока еще допускается использование пространства ввода-вывода, но необходимо учитывать, что его диапазон ограничен и плотно распределен между устройствами современного ПК. Таким образом, в будущем распределение пространства ввода-вывода станет более трудным. Отображение в пространство памяти вместо пространства ввода-вывода дает преимущество, поскольку позволяет использовать устройство даже в тех системах, в которых не поддерживается пространство ввода-вывода. Устройство также может произвольно отобразить его внутренний регистр как в пространство памяти, так и в пространство ввода-вывода, используя два базовых адреса регистра, один для ввода-вывода и другой для памяти. Системное программное обеспечение распределит (если возможно) соответствующее адресное пространство для каждого регистра базового адреса.

Когда вызывается аппаратный драйвер устройства, он определяет, какое адресное пространство должно использоваться для обращения к устройству. Если был бы инициализирован привилегированный механизм доступа в пространство ввода-вывода и регистр базового адреса ввода-вывода, то драйвер обратился бы к устройству, используя транзакции ввода-вывода шины в назначенное адресное пространство ввода-вывода. В противном случае драйвер устройства был бы обязан использовать доступы памяти к адресному пространству, определенному регистром базового адреса памяти. Оба регистра базового адреса обеспечивают доступ к тем же самым регистрам внутренне [9].

ГЛАВА 4



Транзакции

В общем случае под *протоколом обмена* понимается набор правил для организации взаимодействия каких-либо объектов. Спецификация PCI не дает четкого определения протокола обмена между устройствами PCI. Тем не менее в спецификации описывается набор правил и ограничений взаимодействия, требования для объектов-устройств и компонентов PCI, т. е. описывается полноценный протокол шины PCI. В это понятие входит адресация, правила передачи и порядок следования транзакций, арбитраж и т. д. В данной главе рассматриваются транзакции на шине, их разновидности. Описаны типы завершения транзакций, условия завершения, рассмотрен механизм задержанных или отложенных транзакций [9].

Введение

Основным механизмом передачи данных на шине PCI является *блочный механизм*. *Блок* состоит из фазы адреса и одной или более фаз данных. PCI поддерживает передачу блоками как для пространства адресов памяти, так и для пространства адресов ввода-вывода. Данный механизм по сути является пакетным и называется *Burst-режим*. В обычном режиме на каждое считываемое или записываемое слово выдается отдельный адрес, в Burst-режиме адрес выдается на весь пакет данных, затем без задержек непрерывно выполняется серия циклов чтения/записи, что и делает пакетный режим максимально эффективным. Совокупность фазы адреса и одной или нескольких фаз данных называется *транзакцией*. Транзакция происходит между мастером и целью, в направлении от мастера. *Мастером* называется логический узел устройства, выполняющий функции генерирования транзакций, *целью* — логический узел устройства, выполняющий функции конечного объекта транзакции. Устройство может быть многофункциональным, при этом содержать в себе как

функцию (логический узел) мастера, так и цели. Поэтому термин "целевое устройство" применим для однофункционального устройства, для многофункционального этот термин означает одно из нескольких устройств, входящих в состав другого устройства. Мастер и цель представляют собой конечные автоматы. В каждый момент времени на шине может происходить только одна транзакция, поэтому необходимо определить ряд ограничивающих требований. Максимальная продолжительность каждой транзакции составляет 16 тактов. Также для повышения производительности предусмотрено несколько механизмов: *отложенные транзакции* — для обмена с медленными устройствами, *буферизация транзакций* — сохранение транзакции во временном буфере. Поэтому возникает необходимость определения порядка следования и очередности транзакций для того, чтобы избежать коллизий, потерь данных и тупиковых состояний. Одним из методов повышения производительности является снижение числа транзакций. Достигается это за счет объединения нескольких транзакций в одну, слияние байтов и слов в двойные слова и свертка байтов. Для обращения к конкретному байту внутри двойного слова используются линии C/BE#, функционирующие в качестве сигнала разрешения обращения к байтам в фазе данных. Некоторые сигналы могут выставляться разными устройствами, поэтому для передачи управления сигналами применяется оборотный цикл. Взаимодействие двух мастеров описывается с помощью модели "производитель — потребитель" [9].

Общее управление передачей информации

Шина PCI является синхронной, частота тактовых синхроимпульсов задается тактовым генератором. То есть все события на шине происходят по синхросигналам. Изменения состояний сигналов на линиях происходят одновременно с наступлением (формированием) переднего фронта синхроимпульса[1]. Каждый сигнал имеет период установления и интервал удержания относительно переднего фронта синхроимпульса, на котором не разрешены переходы в другое состояние. Вне этого интервала, значения сигнала или переходы не имеют никакого значения. Этот интервал имеет значение только на заданных фронтах синхроимпульсов для сигналов AD[31::00], AD[63::32], PAR[2], PAR64 и IDSEL, и на любом фронте синхроимпульса для сигналов LOCK#, IRDY#, TRDY#, FRAME#, DEVSEL#, STOP#, REQ#, GNT#, REQ64#, ACK#64, SERR# (только по спаду уровня для сигнала SERR#) и PERR#. Для сигналов C/BE[3::0]#, C/BE[7::4]# (при их использовании в качестве команд шины) заданным является первый передний фронт, на котором выставлен

[1] Исключением являются сигналы RST#, INTA#, INTB#, INTC# и INTD#.

[2] Сигналы PAR и PAR64 обрабатываются так же, как и AD, с задержкой на один такт.

сигнал FRAME#. Для сигналов C/BE[3::0]#, C/BE[7::4]# (при их использовании в качестве сигнала разрешения обращения к байтам) заданным является любой передний фронт синхроимпульса после завершения фазы адреса или фазы данных, эти сигналы остаются действительными во время всей фазы данных. Сигналы RST#, INTA#, INTB#, INTC# и INTD# асинхронны.

Передача данных должна быть согласована между передающим и принимающим устройством, необходимо отметить начало и конец транзакции, готовность обоих устройств. На шине PCI для этих целей используются три интерфейсных сигнала:

- FRAME# — отмечает начало и конец транзакции, выставляется мастером;
- IRDY# — указывает готовность мастера к передаче данных;
- TRDY# — указывает готовность цели к передаче данных.

Состояние интерфейса, при котором сигналы FRAME# и IRDY находятся в неактивном состоянии, называется *свободным*. Транзакция начинается с перевода сигнала FRAME# в активное состояние, одновременно начинается фаза адреса, в которой передаются адрес и команда шины. По следующему фронту синхроимпульса начинается первая фаза данных, в которой передаются данные между мастером и целью. Передача происходит по переднему фронту синхроимпульса при условии активности сигналов IRDY# и TRDY#. Циклы ожидания могут быть инициированы в фазе данных мастером либо целью, с помощью снятия сигналов IRDY# и TRDY# соответственно.

Источник данных должен выставить сигнал xRDY# безусловно, если данные корректны (IRDY# — для транзакции записи, TRDY# — для транзакции чтения). Принимающий агент может задержать установку его сигнала xRDY#, если не готов принять данные. При задержке установки сигнала xRDY#, цель и мастер должны ожидать определенное число тактов, называемое *временем ожидания*. Данные передаются только тогда, когда сигналы IRDY# и TRDY# переведены в активное состояние на одном и том же переднем фронте синхроимпульса.

Если мастер выставил сигнал IRDY#, то он не может изменить состояние сигналов IRDY# или FRAME# до завершения текущей фазы данных, независимо от состояния сигнала TRDY#. Если цель выставила сигнал TRDY# или STOP#, то она не может изменять состояние сигналов DEVSEL#, TRDY# или STOP# до завершения текущей фазы данных. Ни мастер, ни цель не могут изменять свое состояние до завершения передачи данных (фаза данных выполняется, когда выставлены сигналы IRDY# и один из сигналов TRDY# или STOP#). Данные передаются в зависимости от состояния сигнала TRDY#.

Если мастер собирается выполнять только одну фазу данных (это может произойти сразу после фазы адреса), то он переводит сигнал FRAME# в неактив-

ное, а сигнал IRDY# — в активное состояние. После того как цель показывает, что готова выполнить заключительную передачу данных (выставлением сигнала TRDY#), интерфейс возвращается в свободное состояние переводом сигналов FRAME# и IRDY# в неактивное состояние [9].

Транзакции чтения и записи

Взаимодействия на шине между устройствами осуществляются с помощью транзакций. Под *транзакцией* в общем случае понимается объединенная по смыслу последовательность фазы адреса и фазы данных, указываемая служебными и управляющими сигналами. Количество фаз данных может изменяться, в одной транзакции может быть одна или несколько фаз данных. Поэтому транзакции подразделяют на:

- одиночные;
- блочные.

В зависимости от совершаемых операций транзакции могут быть следующих типов:

- транзакции чтения;
- транзакции записи;
- конфигурационные транзакции.

Транзакции могут генерироваться только мастером. Разрядность транзакции определяется разрядностью используемой шины адреса/данных. В зависимости от этого транзакции могут быть 32- или 64-разрядными.

Далее описываются основные виды транзакций: чтения и записи. Для системы обозначений, применяемых в иллюстрациях, необходимо дать некоторые пояснения. Сплошная линия означает, что сигнал в данный момент управляется текущим мастером или целевым устройством. Пунктирная линия обозначает отсутствие управления сигналом. Пока пунктирная линия находится в верхнем положении, сигнал может принимать постоянное значение. На диаграммах также обозначено состояние тристабильных выводов, сигналы которых имеют неопределенное значение, когда пунктирная линия находится между двумя положениями (например, линии адреса AD или линиями C/BE#). Переход сплошной линии в пунктирную означает, что сигнал, который активно управлялся, перешел в третье состояние. Переход сплошной линии из нижнего положения в верхнее, а затем в пунктир показывает, что сигнал активно управлялся по высокому уровню до установки на шине, и затем перешел в третье состояние [9].

Транзакция чтения

Рассмотрим подробно, что происходит на каждом такте транзакции чтения, представленной на рис. 4.1. На такте 1 мастер выставляет сигнал FRAME#, который не снимается до конца транзакции. Установка этого сигнала обозначает начало транзакции. По достижении сигналом FRAME# половины заданного уровня мастер начинает переводить в активное состояние сигналы AD и C/BE[3::0]#. Таким образом, формируется фаза адреса, во время которой по линиям AD[31::00] передается адрес, а по линиям C/BE[3::0]# — команда шины.

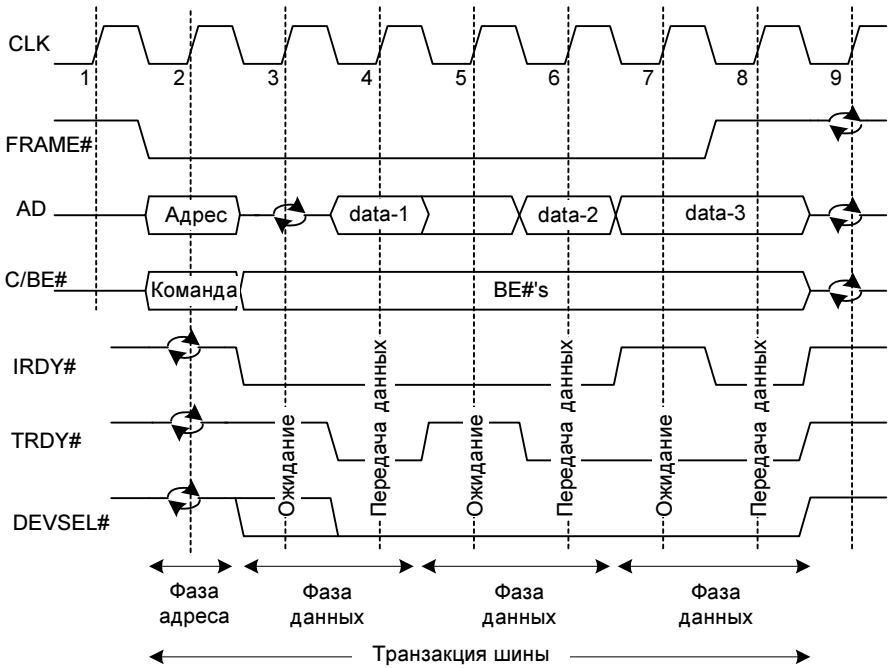


Рис. 4.1. Стандартная транзакция чтения

Далее, на такте 2, происходит выбор устройства. Каждая цель дешифрирует адрес и сравнивает его со значением, хранящимся в регистре базового адреса. Цель, к которой обращается мастер, выставляет сигнал DEVSEL#. Мастер выставляет сигнал IRDY#, указывая готовность к приему. Но т. к. цель еще не готова к передаче данных, то она задерживает формирование сигнала TRDY#. Во время этой задержки происходит оборотный цикл и цель получает управление линиями AD. Все это время мастер ожидает готовности цели. На такте 4 цель выставляет сигнал TRDY#, указывая на готовность к передаче данных, и начинается первая фаза данных. Во время одной фазы данных

передается двойное слово, при этом сигнал C/BE# показывает, какие из 4-х байтов этого двойного слова следует обрабатывать/принимать мастеру. Фаза данных может состоять из циклов передачи данных и циклов ожидания. На тактах 3, 5 и 7 цель вставляет цикл ожидания, при этом сигналы на линиях AD остаются в активном состоянии. На такте 6 мастер вставляет состояние ожидания. Данные передаются на тактах 4, 6, и 8. На такте 7 мастер снимает сигнал FRAME#, указывая завершение транзакции. Заключительная фаза данных заканчивается на такте 8. На этом такте мастер снимает сигнал IRDY#, цель снимает сигналы TRDY# и DEVSEL#. На такте 9 для сигналов FRAME#, AD, C/BE# выполняются оборотные циклы, чтобы передать управление этими сигналами другому мастеру.

Как показано на диаграмме, данные пересылаются в тактах 4, 6 и 8, а на тактах 3, 5 и 7 вставлены циклы ожидания. Первая фаза данных завершается для транзакции чтения за минимальное время. Вторая фаза данных продлевается на такте 5, т. к. неактивен сигнал TRDY#. Последняя же фаза данных увеличена из-за того, что сигнал IRDY# был неактивен на такте 7.

На такте 7 мастеру уже известно, что следующая фаза данных — последняя. Тем не менее сигнал FRAME# остается активным, т. к. мастер не готов завершить последнюю пересылку (сигнал IRDY# на такте 7 снят). Только после установки сигнала IRDY# мастер может снять сигнал FRAME#, что и происходит в такте 8, тем самым указывая цели, что это последняя фаза данных в транзакции. Вся транзакция заняла 8 тактов, это была блочная транзакция, за 1 такт был передан адрес и команда шины, целью было вставлено 2 цикла ожидания, один цикл ожидания был вставлен мастером, с учетом вышеперечисленного данные были переданы за 6 тактов.

Выходные буферы линий C/BE# должны оставаться доступными для чтения и записи от первого такта фазы данных и до конца транзакции. Это правило гарантирует, что сигналы на линиях C/BE# не будут находиться в плавающем состоянии, если необходимо длительное время поддерживать стабильное состояние. Сигнал разрешения обращения к байтам передается в течение всей фазы данных, независимо от состояния линии IRDY#.

Сигнал разрешения обращения к байтам содержит информацию для фазы данных под номером N+1 на такте, следующем после завершения фазы данных номер N. Это не показано на рис. 4.1, потому что в блоке транзакции чтения обычно выставлены все сигналы разрешения обращения к байтам; такое состояние показано на рис. 4.2. На такте 5 рис. 4.2 мастер вставил состояние ожидания снятием сигнала IRDY#. Тем не менее сигнал разрешения обращения к байтам для фазы данных 3 действителен на такте 5 и остается действительным, пока фаза данных не заканчивается на такте 8. Для транзакции чтения первой фазе данных требуется оборотный цикл (инициированный це-

левым устройством установкой сигнала TRDY#). В этом случае на адресных линиях на такте 2 будет действительный адрес, а затем мастер завершит управление адресными линиями. Первый такт, на котором целевое устройство может предоставить данные — это такт 4. Целевое устройство должно управлять адресными линиями после оборотного цикла, когда установлен сигнал DEVSEL#. По возможности, буферы вывода должны оставаться доступными до конца транзакции (для гарантии того, что адресные линии не будут изменяться в течение длительного времени).

Одним из возможных путей для завершения фазы данных является установка сигналов IRDY# и TRDY# на том же фронте синхроимпульса, на котором завершается передача данных. Сигналом TRDY# нельзя управлять, пока установлен DEVSEL#. Когда сняты сигналы IRDY# или TRDY#, то вставляется цикл ожидания, и данные не передаются [9, 18].

Транзакция записи

На рис. 4.2 представлена транзакция записи. Транзакция начинается на такте 2 установкой сигнала FRAME#. Транзакция записи подобна транзакции чтения, за исключением того, что после фазы адреса не требуется оборотный

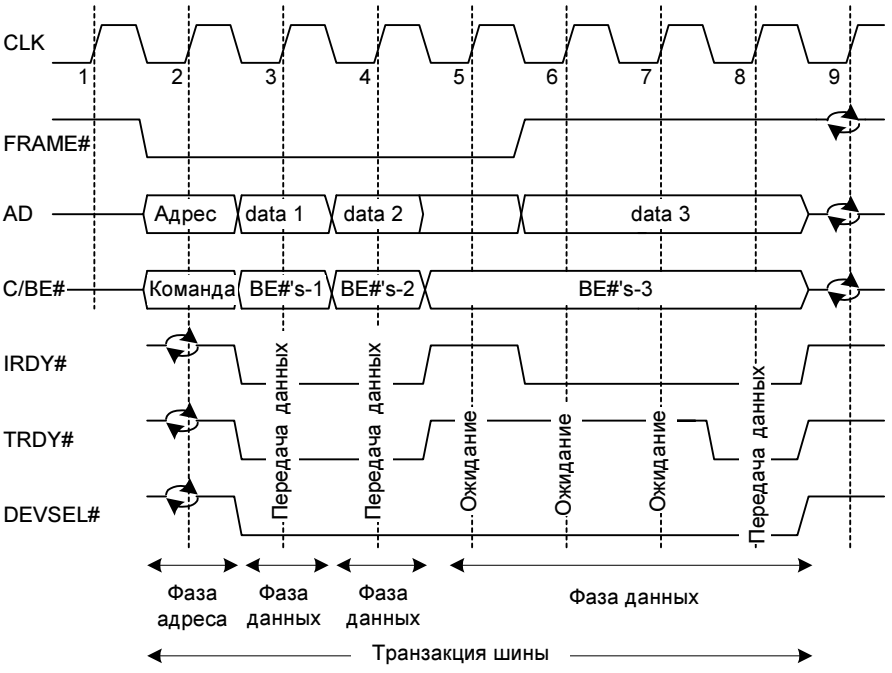


Рис. 4.2. Транзакция записи

цикл, т. к. мастер будет передавать и адрес, и данные. Фазы данных транзакции записи идентичны фазам данных транзакции чтения.

Первая и вторая фазы данных завершаются без циклов ожидания. Однако на третьей фазе данных целью вставлены три цикла ожидания.

Мастер и цель вставляют цикл ожидания на такте 5. Последняя фаза данных указывается активным состоянием сигнала IRDY# при неактивном состоянии сигнала FRAME#. На такте 5 мастер снял сигнал IRDY#, задержав передачу данных. На тактах 5, 6 и 7 цель сняла сигнал TRDY# в связи с неготовностью к приему данных. Последняя фаза данных указывается мастером в такте 6, но она не завершается до такта 8.

Мастер мог бы задержать передачу данных, повременив с установкой сигнала разрешения обращения к байтам, но такая операция запрещена [9, 18].

Завершение транзакции

Мастер обладает исключительным правом начать транзакцию. Но необходимо предусмотреть нештатную ситуацию, в которой продолжение транзакции может привести к краху системы или неработоспособности устройства. Следовательно, правом завершения транзакции должны обладать оба участника обмена, мастер и цель. Остановить транзакцию односторонне, т. е. оборвать ее в один момент времени фактически нельзя, поэтому управление служебными линиями мастер завершает постепенно.

Несмотря на то, что цель может требовать завершения транзакции, правом управления в любом случае обладает мастер. Мастер должен привести все транзакции к завершению надлежащим образом в соответствии с правилами завершения транзакций независимо от того, что именно вызвало нештатное завершение.

Транзакция является *завершенной*, когда неактивны оба сигнала FRAME# и IRDY#, при этом их неактивность указывает состояние "свободно" на шине (например, такт 9 на рис. 4.2).

Завершение транзакции мастером

Для завершения транзакции мастер использует следующий механизм. Сигнал FRAME# переводится в неактивное состояние, а сигнал IRDY# в активное, что и означает завершение транзакции. Совокупность состояний данных сигналов указывает цели, что следующая фаза данных является заключительной. Последняя передача данных происходит, когда активны оба сигнала IRDY# и TRDY#. Транзакция завершается, когда оба сигнала FRAME# и IRDY# неактивны (шина при этом ожидает завершения транзакции).

Существует три следующие причины, по которым мастер завершает транзакцию, используя данный механизм:

- ❒ Нормальное завершение (Completion)
- ❒ Завершение по тайм-ауту (Time-out)
- ❒ Аварийное завершение (Master-Abort)

Нормальное завершение (Completion). Нормальное завершение транзакции осуществляется по окончании передачи данных.

Завершение по тайм-ауту (Time-out). Снят сигнал GNT# мастера, и истекло время ожидания по внутреннему таймеру. Соответствующая транзакция не обязательно должна завершиться. Время может закончиться из-за ожидания ответа цели, либо из-за длительности соответствующей операции. Например, транзакция "Memory Write and Invalidate" не зависит от значения таймера ожидания, кроме как на границе строки кэш-линии. Мастер, инициирующий транзакцию командой *Memory Write and Invalidate*, игнорирует таймер ожидания, пока не будет достигнута граница строки кэш-линии. Когда же это произойдет, и истечет время ожидания (и будет снят сигнал GNT#), мастер должен завершить транзакцию.

Модифицированная версия данного механизма завершения позволяет мастеру завершать транзакцию, когда не отвечает ни одно целевое устройство. Такое завершение называется *"Master-Abort"* — аварийное завершение, инициированное мастером. Хотя завершение "Master-Abort" может вызывать фатальную ошибку для приложения, изначально запросившего транзакцию, это не приводит к общей ошибке системы, обеспечивая, таким образом, нормальную работу шины PCI для других агентов.

На рис. 4.3 показаны два примера нормального завершения транзакции. Признаком заключительной фазы данных является снятие сигнала FRAME# и выставление сигнала IRDY#. Передача данных заключительной фазы происходит, когда неактивен сигнал FRAME#, и активны оба сигнала IRDY# и TRDY#. Шина переходит в свободное состояние по снятию сигнала IRDY#, это происходит на такте 4. Поскольку транзакция завершена, то на такте 4 также снят сигнал TRDY#. На такте 3 допускается нахождение сигнала TRDY# в неактивном состоянии, в этом случае должна обеспечиваться задержка заключительной фазы данных (следовательно, задержка окончания транзакции), пока не будет обеспечена готовность. Если цель оставит сигнал TRDY# в неактивном состоянии на такте 3, то мастер должен будет сохранять сигнал IRDY# активным, пока не завершится заключительная передача данных.

На обеих частях рис. 4.3 ситуация могла бы завершиться и по тайм-ауту (по истечении времени). Как показано на левой части рисунка, сигнал FRAME#

неактивен в такте 3, т. к. истекает время по таймеру. Сигнал GNT# переходит в неактивное состояние, и мастер готов (сигнал IRDY# активен) к заключительной передаче. Поскольку сигнал GNT# неактивен по истечении времени, то дальнейшее использование шины не разрешается, за исключением случая использования команды *Memory Write and Invalidate*, которая должна завершиться при достижении границы строки кэша. Далее завершение происходит нормальным образом. Если сигнал TRDY# снят на такте 2, то фаза данных будет продолжаться, пока он не перейдет в активное состояние. Сигналы FRAME# и IRDY# должны оставаться активными, пока не завершится фаза данных.

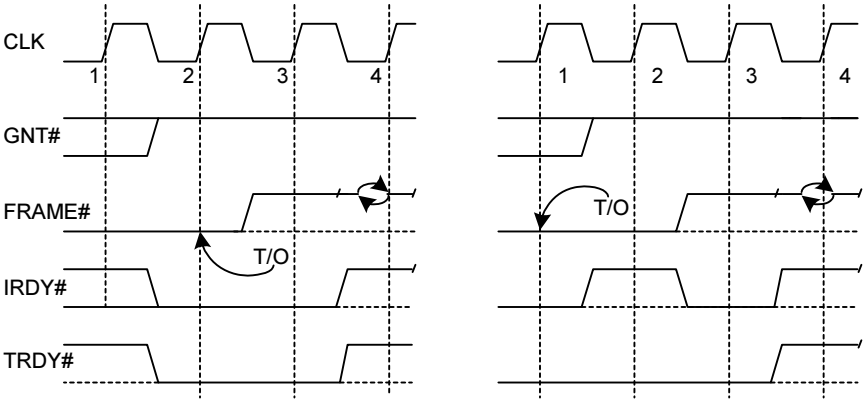


Рис. 4.3. Завершение транзакции мастером

На правой части рисунка показана ситуация, когда время истекает на такте 1. Поскольку мастер не готов к передаче данных (сигнал IRDY# снят на такте 2), то сигнал FRAME# должен оставаться активным. На такте 3 сигнал FRAME# снят, т. к. мастер готов (выставлен сигнал IRDY#) к завершению транзакции на такте 3. Такая задержка завершения не должна превышать 2 или 3 такта. Кроме этого, транзакция не должна завершаться по истечении времени до тех пор, пока не будет снят сигнал GNT#. Мастер всегда должен передавать данные (при записи), либо быть способным к получению данных (при чтении), когда активен сигнал IRDY#.

Аварийное завершение (Master-Abort). Завершение "Master-Abort", показанное на рис. 4.4 , является аварийным случаем завершения (кроме транзакций конфигурации или команд специального цикла), инициированного мастером. Мастер определяет, что на запрошенную транзакцию не будет ответа, если сигнал DEVSEL# останется неактивным на такте 6. Это может происходить, если целевое устройство неспособно к работе, или был передан неправильный адрес, в этом случае мастер не должен повторять транзакцию. Как

только мастер обнаружил отсутствие сигнала DEVSEL# (такт 6 в данном примере), он переводит сигнал FRAME# в неактивное состояние на такте 7, а сигнал IRDY# — на такте 8. Мастер может завершить транзакцию аварийным способом не раньше, чем через 5 тактов после первой установки сигнала FRAME#. Данное условие применяется для одиночной транзакции. Если транзакция блочная, то она может занимать больше пяти тактов и мастеру может потребоваться больше времени для снятия сигнала FRAME# и завершения запроса. Мастер должен поддерживать связь состояний сигналов FRAME# и IRDY# во всех транзакциях, которые включают в себя аварийное прекращение работы. *Сигнал FRAME# может быть снят только во время активности сигнала IRDY#, а сигнал IRDY# должен находиться в установленном состоянии, по крайней мере, один такт после того, как FRAME# станет неактивным, даже когда транзакция заканчивается с аварийным завершением.*

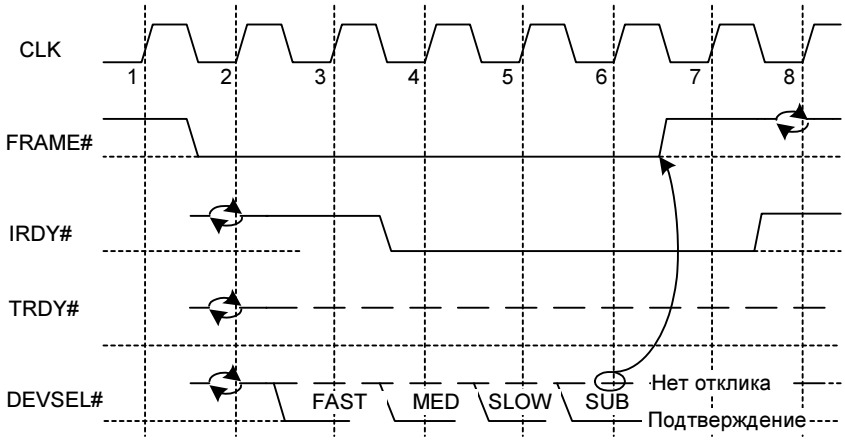


Рис. 4.4. Завершение транзакции "Master-Abort"

В качестве альтернативы, сигнал IRDY# мог бы стать неактивным на такте 7 при условии, что FRAME# был снят, как в случае транзакции с одной фазой данных. В нормальной ситуации мастер не будет повторять данный запрос. Если бы сигнал DEVSEL# был выставлен на тактах 3, 4, 5 или 6 в данном примере, то, следовательно, агент ответил на запрос и аварийное завершение было бы недопустимо.

Главный мост в PC-совместимых системах должен возвратить все единицы для транзакции чтения и отменить данные для транзакции записи, когда происходит аварийное завершение. В регистре состояния моста должен быть установлен бит **Master-Abort**. Другие мастер-устройства могут сообщать об этом состоянии как об ошибке с помощью сигнала SERR#, при отсутствии

возможности сообщить об ошибке через драйвер устройства. Если мост PCI-to-PCI используется в другой системе, то он должен сообщить об ошибке так же, как и другие мастера. Упреждающее чтение данных не должно влиять на систему. Это означает, что когда упреждающая транзакция завершена с аварийным завершением "Master-Abort", то мост должен просто остановить транзакцию и продолжать нормальную работу без каких-либо сообщений [9].

Таким образом, перечисленные требования и рекомендации в обобщенном виде представляют собой общие правила управления сигналами FRAME# и IRDY# во всех транзакциях PCI.

- Сигналы FRAME# и IRDY# определяют состояние шины занято/свободно (BUSY/IDLE); активность этих сигналов означает занятость шины (BUSY); неактивное состояние обоих сигналов означает, что шина находится в свободном состоянии (IDLE).
- Сигнал FRAME# может переходить в неактивное состояние только один раз для каждой транзакции.
- Сигнал FRAME# не может быть переведен в неактивное состояние, если выставлен сигнал IRDY# (IRDY# должен всегда находиться в активном состоянии на первом после перевода в неактивное состояние сигнала FRAME# фронте синхроимпульса).
- Как только мастер перевел сигнал IRDY# в активное состояние, он не должен изменять состояние сигналов IRDY# или FRAME# до окончания текущей фазы данных.
- Мастер должен снять сигнал IRDY# один такт спустя после завершения последней фазы данных.

Завершение транзакции целью

В большинстве случаев цель способна к получению или передаче данных до завершения транзакции мастером. При некоторых условиях цель не в состоянии закончить запрос, поэтому для завершения транзакции она может использовать сигнал STOP#. Комбинация сигнала STOP# с другими сигналами указывает мастеру на условия завершения. Цели могут инициировать три типа завершения:

- Повтор (Retry)
- Отключение (Disconnect)
- Аварийное завершение (Target-Abort)

Повтор (Retry). Завершение, требуемое до передачи каких-либо данных, потому что цель занята и временно не в состоянии обработать транзакцию. Та-

кая ситуация может возникнуть, например, если устройство не может выполнить требование начального времени ожидания, в настоящее время занята другим мастером, или возник конфликт внутреннего ресурса. "Retry" — специальный случай завершения "Disconnect without Data", происходящий на начальной фазе данных. Цель указывает "Retry", выставляя сигнал STOP# и не устанавливая TRDY# на начальной фазе данных транзакции (STOP# не может быть выставлен во время оборотного цикла между фазой адреса и первой фазой данных транзакции чтения). При использовании целью завершение "Retry" данные не передаются.

Отключение (Disconnect). Завершение, которое происходит из-за неспособности целью продолжать блочную передачу, одновременно с передачей данных или после нее во время начальной фазы данных. Вызвано неспособностью цели ответить в пределах требования времени ожидания, например, вследствие пересечения блоком границы ресурса или конфликта ресурса. Данный вид завершения может происходить с передачей данных или без нее. Завершение "Disconnect with Data" может быть сообщено на любой фазе данных установкой сигналов TRDY# и STOP# одновременно. Это завершение используется, когда цели необходимо завершить только текущую фазу данных. Завершение "Disconnect without Data" может быть сообщено на любой последующей фазе данных (значимые данные были переданы на предыдущей фазе данных) снятием сигнала TRDY# и установкой сигнала STOP#. Завершение "Disconnect" отличается от "Retry" тем, что завершение "Retry" всегда происходит на начальной фазе данных и данные не передаются. Если данные переданы одновременно или до завершения транзакции целью, то это тип завершения "Disconnect". Завершение "Disconnect" может произойти во время начальной фазы данных из-за неспособности цели завершить блок.

Аварийное завершение (Target-Abort). Аварийное завершение, которое происходит из-за обнаружения целью критической ошибки, или из-за неспособности цели закончить запрос. Хотя такое завершение может вызвать фатальную ошибку для устройства, запросившего транзакцию, такое завершение транзакции позволяет работать другим агентам. Например, мастер запрашивает чтение двойного слова из пространства ввода-вывода, но для данной цели доступ в это пространство осуществляется побайтно. Поскольку цель не может выполнить запрос, цель заканчивает запрос с аварийным завершением "Target-Abort". Как только цель подтвердила доступ установкой сигнала DEVSEL#, она может сообщать "Target-Abort" на любых последующих тактах. Аварийное завершение "Target-Abort" сообщается целью снятием сигнала DEVSEL# и установкой сигнала STOP# одновременно.

Способность к завершению типа "Retry" должно реализовать большинство целей, любые другие типы завершения транзакций не являются обязательны-

ми для реализации. Мастера должны поддерживать все типы завершения транзакций целью. Завершение типа "Retry" не является обязательным для самых простых типов целей, которые:

- ❒ не поддерживают монопольные доступы;
- ❒ не имеют буфера отложенной записи в память, который нуждается в очищении для выполнения правил упорядочивания;
- ❒ не могут перейти в состояние, в котором необходимо отклонить доступ;
- ❒ могут всегда ответить в пределах начального времени ожидания.

Цели разрешено сообщить "Disconnect with Data" (выставив сигналы STOP# и TRDY#) на начальной фазе данных, даже если сигнал FRAME# находится в неактивном состоянии [9].

Правила завершения транзакции целью

Следующие общие правила определяют управление сигналами FRAME#, IRDY#, TRDY#, STOP# и DEVSEL# при завершении транзакций.

1. Фаза данных заканчивается на любом переднем фронте синхроимпульса, на котором выставлен сигнал IRDY# и один из сигналов STOP# или TRDY#.
2. Независимо от состояния сигнала STOP#, передача данных имеет место на каждом переднем фронте синхроимпульса, где выставлены оба сигнала IRDY# и TRDY#.
3. Цель может снять сигнал STOP# только после снятия сигнала FRAME#.
4. Если цель выставила сигнал TRDY# или STOP#, то она не может изменить состояние сигналов DEVSEL#, TRDY# или STOP# до завершения текущей фазы данных.
5. Как только выставлен сигнал STOP#, мастер должен снять сигнал FRAME#, при условии активности сигнала IRDY#.
6. Сигналы TRDY#, STOP# и DEVSEL# должны быть сняты на такте, следующем после завершения последней фазы данных. На следующем такте эти сигналы должны быть переведены в третье состояние.

Правило 1 означает, что фаза данных может завершиться с установкой или без сигнала TRDY#. Когда цель неспособна выполнить передачу данных, она может выставить сигнал STOP#, не выставляя сигнал TRDY#. Когда выставлены оба сигнала FRAME# и IRDY#, мастер выполняет две первые фазы данных. Мастер не может снять сигнал FRAME#, пока не завершится текущая фаза данных, потому что выставлен сигнал IRDY#. Поскольку допускается выполнение фаз данных, когда установлены сигналы STOP# и IRDY#, масте-

ру разрешено начать заключительную фазу данных. Для этого необходимо снять сигнал FRAME#, удерживая в активном состоянии сигнал IRDY#. Мастер должен снять сигнал IRDY# на такте, следующем после завершения последней фазы данных.

Правило 2 указывает, что данные передаются независимо от состояния сигнала STOP#, если выставлены оба сигнала TRDY# и IRDY#.

Правило 3 означает, что после установки сигнал STOP# должен оставаться в активном состоянии до конца транзакции. Последняя фаза данных транзакции заканчивается, когда снят сигнал FRAME# и выставлены сигналы IRDY# и STOP# (или TRDY#). Цель должна удерживать установленным сигнал STOP#, пока не снят сигнал FRAME# и выставлен сигнал IRDY# (окончание последней фазы данных). То есть цель не может соотносить по времени установку сигнала STOP# и снятие FRAME#. Сигнал STOP# должен быть снят на такте, следующем после завершения последней фазы данных. Когда сигналы STOP# и TRDY# выставлены в одной фазе данных, цель передаст данные в этой фазе данных. В этом случае сигнал TRDY# должен быть снят по завершению фазы данных. Сигнал STOP# должен остаться установленным до конца транзакции, после чего он снимается. Если цели требуется состояние ожидания в фазе данных, где выставлен сигнал STOP#, она должна задержать установку сигнала STOP#, пока не будет готова закончить фазу данных.

Правило 4 означает, что цели не позволено изменить ее состояние, как только она начала выполнение текущей фазы данных. Выполнение фазы данных происходит, когда цель выставляет или сигнал TRDY#, или STOP#. В зависимости от состояний данных сигналов цель может выполнять следующие операции:

- передачу данных в текущей фазе данных и продолжение транзакции (если она блочная), выставляя сигнал TRDY# и не выставляя сигнал STOP#;
- передачу данных в текущей фазе данных и завершение транзакции установкой сигналов TRDY# и STOP#;
- завершение транзакции установкой сигнала STOP# и снятием сигнала TRDY#. В текущей фазе данных сами данные не передаются;
- завершение транзакции с состоянием ошибки "Target-Abort" путем установки сигнала STOP# и снятия сигналов TRDY# и DEVSEL#. Данные в текущей фазе данных не передаются.

Цель не может выполнить текущую фазу данных, если сняты оба сигнала TRDY# и STOP#, и должна вставить состояние ожидания.

Правило 5 означает, что, когда мастер обнаруживает установку сигнала STOP#, он должен снять сигнал FRAME# на первом такте после того такта, на котором выставлен сигнал IRDY#. Установка сигнала IRDY# и снятие

FRAME# должно произойти как можно быстрее после установки сигнала STOP#, желательно в пределах одного-трех тактов. Данная установка сигнала IRDY# (и, следовательно, снятие сигнала FRAME#) может происходить вследствие нормального управления сигнала IRDY# мастером в незавершенной целью транзакции. Альтернативной является ситуация, если снят сигнал TRDY# (дальнейшей передачи данных не будет). В этом случае мастер может выставлять сигнал IRDY# немедленно (даже если он не готов закончить передачу данных). Если транзакция "Memory Write and Invalidate" завершается целью, мастер заканчивает транзакцию (оставшуюся часть строки кэша) как можно скорее (придерживаясь протокола STOP#), используя команду *Memory Write* (при этом условии выходные данные команды *Memory Write and Invalidate* не действительны).

Правило 6 требует, чтобы цель освободила линии в том же порядке, как если бы транзакция завершалась мастером.

Типы завершения транзакций "Retry" и "Disconnect" являются нормальными. Завершение типа "Target-Abort" является ненормальным завершением, которое, возможно, вызвано ошибкой. Поскольку сообщение об ошибке является необязательным, то шина должна функционировать так же, как если бы ошибка не происходила [9].

Операция "Retry"

Рассмотрим пример завершения транзакции типа "Retry", представленный на рис. 4.5. Транзакция начинается с установки мастером сигнала FRAME# на такте 2 и сигнала IRDY# на такте 3. Мастер запрашивает несколько фаз данных, на что указывает установка сигналов FRAME# и IRDY# на такте 3. Цель требует доступа, выставляя сигнал DEVSEL# на такте 4. Далее цель определяет, что она не может выполнить запрос мастера и на такте 4 выставляет сигнал STOP#, при этом сигнал TRDY# находится в неактивном состоянии. Первая фаза данных заканчивается на такте 4, потому что выставлены сигналы IRDY# и STOP#. Поскольку сигнал TRDY# находился в неактивном состоянии, то во время начальной фазы данные не передавались. Мастер получает информацию, что цель не будет передавать данные в этой транзакции, поскольку на такте 4 был выставлен сигнал STOP#, и сигнал TRDY# находился в неактивном состоянии. Мастер обязан снять сигнал FRAME# при условии, что выставлен сигнал IRDY#. В данном случае сигнал FRAME# снят на такте 5, потому что на этом такте выставлен сигнал IRDY#. Последняя фаза данных завершается на такте 5, потому что снят сигнал FRAME# и выставлен сигнал STOP#. Цель снимает сигналы STOP# и DEVSEL# на такте 6, потому что транзакция завершена. Данная транзакция состояла из двух фаз данных, в которой данные не передавались, и мастер обязан снова повторить запрос [9].

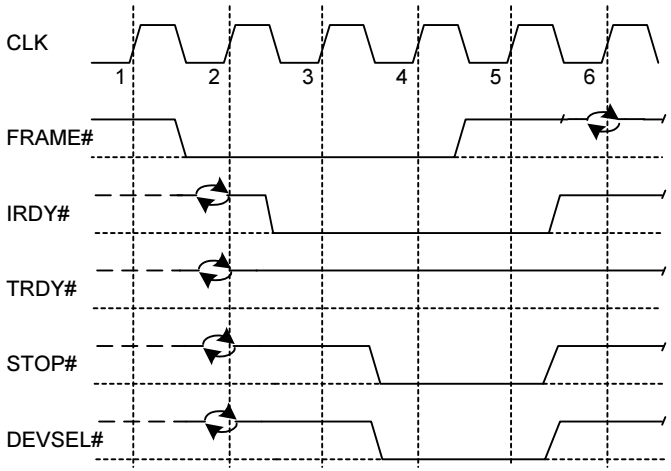


Рис. 4.5. Завершение транзакции типа "Retry"

Операция "Disconnect with Data"

Рассмотрим пример завершения транзакции типа "Disconnect with Data", представленный в двух вариантах. На рис. 4.6 показан вариант "Disconnect-A", где мастер вставляет состояние ожидания, когда цель сообщает "Disconnect with Data". На рис. 4.6 представлена средняя часть транзакции, но для удобства синхроимпульсы нумеруются по порядку. Текущая фаза данных выполняется на такте 3. На тактах 1 и 2 мастер вставил состояние ожидания (снят сигнал IRDY#). На такте 1 цель также не готова к передаче и вставляет состояние ожидания (снят сигнал TRDY#). Так как цель может выполнить только текущую фазу данных, она одновременно выставляет сигналы TRDY# и STOP#. В этом примере данные переданы в последней фазе данных. Поскольку на такте 2 мастер обнаруживает установку сигнала STOP#, то на такте 3 он снимает сигнал FRAME#. Сигнал IRDY# выставлен, поэтому мастер готов выполнить фазу данных. Так как на такте 3 снят сигнал FRAME# и выставлен сигнал STOP#, то последняя фаза данных выполняется. В этой фазе данные передаются, потому что выставлены сигналы IRDY# и TRDY#. Необходимо отметить, что сигнал STOP# выставлен на тактах 2 и 3. Это объясняется тем, что цель обязана удерживать сигнал STOP# установленным, пока не снят сигнал FRAME#. В этом примере данные передаются в последней фазе данных.

Второй вариант завершения транзакции типа "Disconnect With Data" представлен на рис. 4.6 "Disconnect-B". Он совпадает с "Disconnect-A", за исключением того, что в последней фазе данных снят сигнал TRDY#. В этом примере данные были переданы на тактах 1 и 2, но не во время последней фазы

данных. Цель указывает, что она не может продолжить блок, выставляя сигналы STOP# и TRDY#. Поскольку цель не может выполнить третью фазу данных, то на такте 2 цель обязана снять сигнал TRDY# и удерживать сигнал STOP# установленным. Последняя фаза данных выполняется без передачи данных на такте 3, потому что снят сигнал TRDY#, и выставлен сигнал STOP#. В этом примере представлены три фазы данных, в двух из которых данные передаются и в одной не передаются.

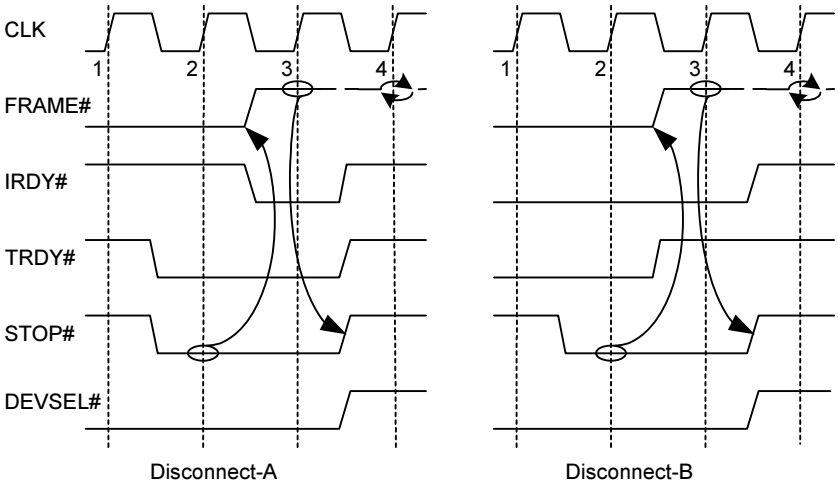


Рис. 4.6. Завершение транзакции типа "Disconnect with Data"

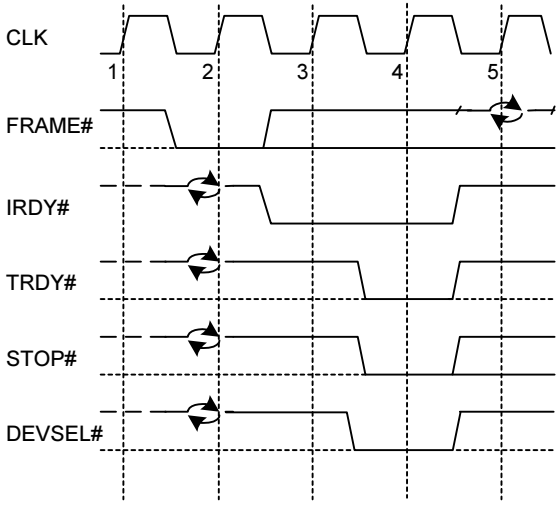


Рис. 4.7. Завершение транзакции мастером

На рис. 4.7 показан пример завершения транзакции мастером, где цель "вслепую" выставляет сигнал STOP#. Данный пример является нормальным завершением одиночной транзакции. Мастер запрашивает транзакцию с единственной фазой данных, а цель "вслепую" выставляет сигналы STOP# и TRDY#, указывая, что она может выполнить только одну фазу данных. Транзакция начинается как обычно с установки сигнала FRAME#. Мастер указывает, что начальная фаза данных является заключительной, потому что на такте 3 снят сигнал FRAME# и выставлен сигнал IRDY#. Цель требует транзакции, указывая, что готова передать данные, и запрашивает прерывание транзакции, выставляя сигналы DEVSEL#, TRDY# и STOP# одновременно. Таким образом, данные переданы в одной фазе данных [9].

Операция "Disconnect without Data"

На рис. 4.8 представлен тип завершения транзакции "Disconnect without Data". Транзакция начинается с установки сигнала FRAME# на такте 2 и установки сигнала IRDY# на такте 3. Мастер запрашивает несколько фаз данных, потому что на такте 3 выставлены оба сигнала FRAME# и IRDY#. На такте 4 цель отвечает на транзакцию, выставляя сигнал DEVSEL#.

На такте 4 выполняется первая фаза данных, на такте 5 — вторая. На такте 6 мастер готов к продолжению блока, о чем говорят выставленные сигналы FRAME# и IRDY#. Но цель не в состоянии продолжить выполнение фаз данных и на такте 6 устанавливает сигнал STOP#, одновременно снимая сигнал TRDY#. Поскольку сигналы IRDY# и STOP# выставлены на такте 6, третья фаза данных выполняется уже без передачи данных. Цель продолжает удерживать сигнал STOP# в активном состоянии на такте 7, потому что на такте 6 выставлен сигнал FRAME#. На такте 7 выполняется четвертая и заключительная фаза данных, т. к. на такте 7 снят сигнал FRAME# (сигнал IRDY# выставлен) и установлен сигнал STOP#. На такте 8 шина возвращается в состояние "свободно". В данном примере в первых двух фазах данных выполняется передача данных, а в последних двух — нет. Такое может произойти, если устройство получило два двойных слова и определило, что его буферы переполнены или если блок пересек границу ресурса, т. е. адрес части блока не принадлежит данному устройству. Цель в состоянии выполнить первые две фазы данных, но не в состоянии выполнить третью. Если мастер инициирует новый запрос с целью продолжения блока, устройство, которое содержит адрес следующих не переданных данных, будет требовать доступа и продолжит блок.

На рис. 4.9 представлен возможный вариант завершения транзакции "Disconnect without Data". Он идентичен варианту, показанному на рис. 4.8, за исключением того, что на такте 6 одновременно с установкой целью сигнала

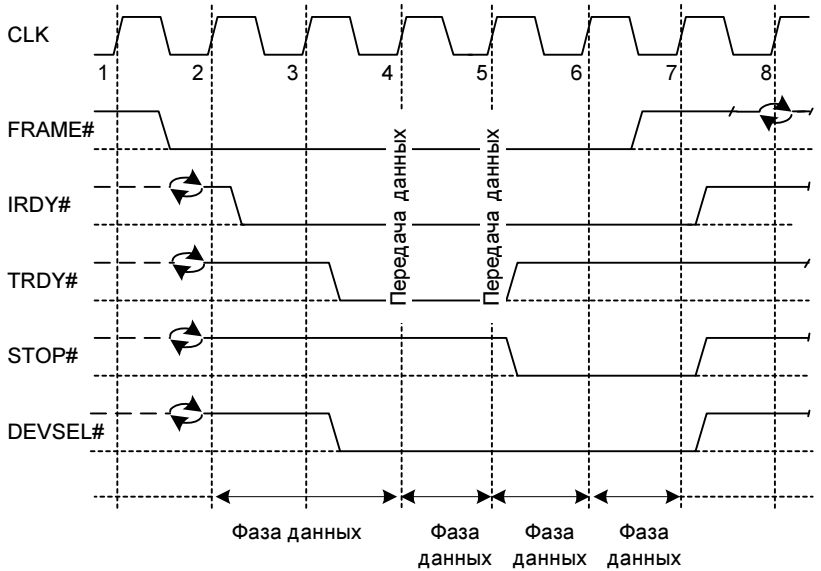


Рис. 4.8. Пример 1 завершения типа "Disconnect without Data"

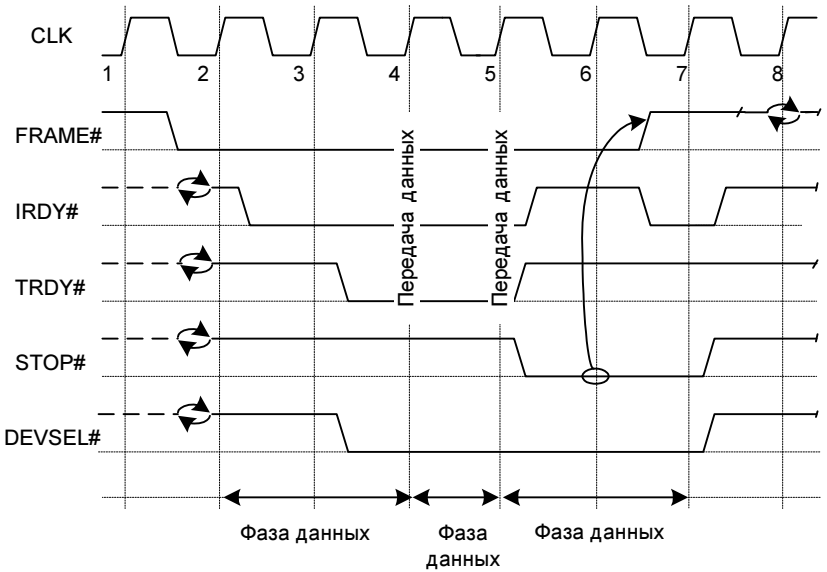


Рис. 4.9. Пример 2 завершения типа "Disconnect without Data"

STOP# и снятием TRDY# мастер вставляет состояние ожидания снятием сигнала IRDY#. Поскольку сигнал FRAME# не был снят на такте 5, мастер предпринял, по крайней мере, еще одну фазу данных и должен выполнить ее. Здесь простым решением было бы, обнаружив выставленный сигнал STOP#, просто снять сигнал FRAME# и завершить транзакцию. Но такая последовательность состояний сигналов FRAME# — IRDY# является нарушением протокола шины. Поэтому мастер снимает сигнал FRAME# и выставляет сигнал IRDY#, указывая, что это последняя фаза данных, которая выполняется на такте 7, т. к. выставлен сигнал STOP#. Данный пример состоит из трех фаз данных, предыдущий имел четыре. Состояние ожидания позволило мастеру выполнить транзакцию с третьей фазой данных [9].

Завершение "Target-Abort"

Завершение транзакции типа "Target-Abort" является аварийным и представлено на рис. 4.10. Завершение "Target-Abort" показывает, что цель требует остановки транзакции и запрещает мастеру повторять запрос, т. к. вообще не в состоянии его выполнить. На рис. 4.10 не показана начальная часть транзакции, но до такта 1 мастер выставил сигнал FRAME#, чтобы начать запрос, и цель ответила установкой сигнала DEVSEL#. До такта 1 также могли присутствовать фазы данных, что не принципиально. Цель определяет, что мастер потребовал транзакцию, которую она неспособна выполнить, или что произошла фатальная ошибка. Прежде чем цель может сообщить "Target-Abort", сигнал DEVSEL# должен быть выставлен для одного или более тактов. К моменту перевода сигнала DEVSEL# в неактивное состояние и

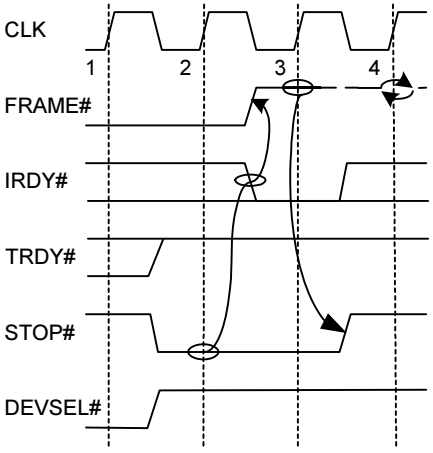


Рис. 4.10. Завершение типа "Target-Abort"

установки сигнала STOP# (что и является завершением "Target-Abort") сигнал TRDY# должен быть снят. Эти сигналы переводятся в соответствующие состояния на такте 2. Данные, переданные в предыдущих фазах данных этой транзакции, возможно, были повреждены. Поскольку на такте 2 установлен сигнал STOP#, и на такте 3 мастер может выставить сигнал IRDY#, то на такте 3 мастер снимает сигнал FRAME#. Транзакция заканчивается на такте 3, потому что выставлены сигналы IRDY# и STOP#. На такте 4 мастер снимает сигнал IRDY# и цель снимает сигнал STOP# [9].

Требования для мастеров

Реализация всех типов завершения транзакций не является обязательной для целей, тем не менее мастера должны быть в состоянии обработать все типы завершения транзакций целью, правила обработки и различные условия приведены далее.

Снятие сигнала REQ#. При завершении транзакции типа "Retry", "Disconnect with Data" или "Disconnect without Data" мастер должен снять сигнал REQ# перед тем, как повторять транзакцию. Устройство, содержащее один логический блок "мастер", должно снять сигнал REQ# как минимум на два такта. Один такт на время перехода шины в состояние "свободно" (в конце транзакции, где был выставлен сигнал STOP#) и один такт до или после состояния "свободно".

Некоторые устройства могут содержать несколько мастеров, которые используют один общий для всех вывод REQ#.

Примерами таких устройств являются:

- однофункциональное устройство, которое содержит две независимых подфункции. Одну, которая производит данные, и одну, которая потребляет данные;
- многофункциональное устройство;
- мост PCI-to-PCI, который способен к перенаправлению многократных задержанных транзакций из другой шины.

Для устройств, содержащих несколько мастеров с одним общим выводом REQ#, предоставляются исключения из правил. Каждый из мастеров может использовать шину (предполагая, что не снят сигнал GNT#) без снятия сигнала REQ#, даже если транзакции одного или более мастеров завершены целью (выставлен сигнал STOP#). Устройство должно снять сигнал REQ# как минимум на 2 такта перед началом любой транзакции, которая была завершена целью и может быть повторена. Один из этих тактов необходим на время, пока шина находится в состоянии "свободно".

Мастер не обязан снимать его сигнал REQ#, если цель требует завершения транзакции, выставляя сигнал STOP# в последней фазе данных. Примером

является тип завершения, представленный на рис. 4.7, где транзакция завершена именно мастером, а не целью.

Повторение запроса, завершенного с типом "Retry". Мастер должен повторять запрос, завершенный целью с типом завершения "Retry", пока он не выполнится, но не обязан повторять транзакцию при завершении типа "Disconnect". Запрос необходимо повторить в точности. Это означает, что при повторном доступе должны использоваться тот же адрес (включая состояние линий AD[1::0]), та же команда, тот же сигнал разрешения обращения к байтам, те же данные для записи в случае транзакции записи, и, если поддерживаются, сигналы LOCK# и REQ64#, которые использовались в первоначальном запросе. При этом мастер должен повторить данную транзакцию независимо от любых последующих событий (кроме отмеченных далее), пока первоначальная транзакция не будет выполнена. В то же время, от мастера не требуется немедленного повторения транзакции. В самом простом случае, мастер запросил бы использование шины спустя два такта после того, как был снят сигнал REQ# и повторил бы ту же самую транзакцию. Также мастеру разрешено выполнять другие транзакции шины, но не требуется, чтобы они были выполнены перед повторением первоначальной транзакции. Если устройство реализует и мастера и цель, оно, кроме всего прочего, должно быть способно к приему транзакции. Хорошим примером является многофункциональное устройство. Функции 1, 2, и 3 из одного устройства запрашивают использование интерфейса. Функция 1 запрашивает транзакцию чтения и заканчивается с "Retry". Как только функция 1 перевела шину в состояние "свободно", функция 2 может предпринимать транзакцию в предположении, что для данного устройства сигнал GNT# является активным. После того как функция 2 освобождает шину, может выполняться функция 3, если не снят сигнал GNT#. Как только функция 3 выполнена, устройство должно снять его сигнал REQ# на два такта перед тем, как выставить снова. Как показано выше, функция 1 не обязана завершать ее транзакцию до того, как другая функция может запросить транзакцию. Однако функция 1 должна повторить ее доступ независимо от типа завершения транзакций функций 2 или 3. Условие обязательного повторения транзакции мастером означает, что повторение транзакции не может быть отменено любым другим событием или состоянием. Данное правило относится ко всем транзакциям, завершенным типом "Retry" и не зависит от количества предыдущих транзакций, завершенных с "Retry". Если бы в рассмотренном примере транзакция функции 2 была бы завершена с типом "Retry", то функция 2 должна была бы безоговорочно повторить ее, как и функция 1. Функция 1 и функция 2 должны быть в состоянии повторить их первоначальные запросы независимо от типа завершения транзакции другой функции или транзакции функции 3. Следующие транзакции функции 1 или 2 могли завершиться установкой сигналов SERR#,

PERR#, или закончиться с типом завершения "Retry", "Disconnect", "Target-Abort" или "Master-Abort". Ни одно из этих событий не повлияло бы на требование повторения запроса, который был закончен с типом "Retry".

Мастер должен повторить транзакцию, завершенную с типом "Retry" как можно раньше, желательно в пределах 33 тактов. Однако, при некоторых состояниях, мастер неспособен повторить запрос. Эти состояния обычно возникают вследствие ошибки; например, система выставляет сигнал RST#, драйвер устройства сбрасывается, и затем происходит инициализация компонента, либо программное обеспечение отключает мастера путем сбрасывания бита **Master Bus** (бит 2 в регистре COMMAND). Когда мастер повторяет транзакцию и передает данные, он не обязан продолжать транзакцию после первой фазы данных.

Правила указывают, что перед тем, как повторять первоначальную транзакцию, мастер может выполнить другие транзакции шины. Но в правилах не указано, что эти транзакции должны быть обязательно выполнены перед повторением первоначальной транзакции (которая была завершена целью с типом завершения "Retry"). Если мастер нарушает данное правило, т. е. пытается выполнить транзакции перед повторением первоначального запроса, то это может привести к временным задержкам. Такие задержки, или блокировки, значительно влияют на производительность устройства и системы. Подобным образом могут функционировать устройства, разработанные для предыдущих версий спецификации шины PCI. Временные блокировки должны завершиться по истечении таймера [9].

Задержанные транзакции

Общее быстродействие системы определяется самым медленным элементом. Система с элементами, работающими с одинаковой скоростью на практике, трудноосуществима. Следовательно, необходимо заранее предусмотреть возможность присутствия на шине медленных устройств и механизм обмена с ними. Такой механизм для шины PCI разработан и носит название *механизма задержанных*, или *отложенных транзакций*. Суть его состоит в следующем. Обмен с медленным устройством происходит в несколько этапов, т. к. при непрерывном обмене вся шина ждала бы, пока оно обработает и выполнит запрос. Первый запрос указывает, что должна выполнить цель, во время второго происходит передача данных и затем завершение обмена.

Существует два типа устройств, реализующих задержанные транзакции: контроллеры ввода-вывода и мосты (в частности, мосты PCI-to-PCI). Контроллеры ввода-вывода могут обрабатывать только одну задержанную транзакцию в один момент времени. Мосты могут обрабатывать несколько транзакций в целях повышения производительности системы. Основным преимуществом

задержанных транзакций является то, что шина не находится в состоянии занятости при выполнении доступа к медленному устройству. Мастер, инициировавший такую транзакцию, проходит алгоритм арбитража на шине и уступает ее. В это время другие мастера могут использовать шину, которая в противном случае была бы занята, пока мастер ожидает ответа устройства. Другое преимущество состоит в том, что все буферизованные данные для записи в память не должны быть очищены прежде, чем принят запрос. Фактически очищение буферизованных данных для записи в память происходит прежде, чем задержанная транзакция заканчивается на исходной шине. Это обуславливает возможность использования буферизации во время выполнения небуферизованной транзакции и при этом позволяет не нарушить упорядочивающие правила системы.

Далее рассмотрена реализация механизма задержанных транзакций и требования к устройству, которое может выполнять одну задержанную транзакцию в один момент времени, а также поддержку многократных задержанных транзакций [9].

Алгоритм задержанной транзакции

Все команды шины, которые должны быть выполнены на шине предназначения до выполнения на исходной шине, могут быть исполнены во время задержанной транзакции. К этим командам относятся: *Interrupt Acknowledge*, *I/O Read*, *I/O Write*, *Configuration Read*, *Configuration Write*, *Memory Read*, *Memory Read Line*, и *Memory Read Multiple*.

Команды *Memory Write* и *Memory Write and Invalidate* могут выполниться на исходной шине раньше, чем на шине предназначения (т. е. могут быть буферизованы). Ни одна команда не может быть выполнена путем использования завершения задержанной транзакции и либо буферизуется, либо завершается с типом "Retry". Для контроллеров ввода-вывода термин *шина предназначения* обозначает внутреннюю шину, содержащую ресурс, к которому адресована транзакция. Для моста шина предназначения означает интерфейс, действующий в качестве цели. Например, вторичная шина моста является шиной предназначения, когда транзакция возникает на основной шине моста и адресована цели устройства, находящегося на вторичной шине моста. И наоборот, транзакция, перемещаемая в противоположном направлении, имела бы в качестве шины предназначения первичную шину.

Задержанная транзакция выполняется за три операции:

1. Запрос мастера.
2. Выполнение запроса целью.
3. Выполнение транзакции мастером.

Первая фаза. Мастер генерирует транзакцию на шине, цель дешифрирует доступ, фиксирует информацию, требуемую для выполнения доступа, и завершает запрос с типом "Retry". Зафиксированная информация запроса называется *задержанный запрос*. При завершении запроса с типом "Retry" не существует признаков, по которым мастер мог бы отличить цель, которая заканчивает транзакцию, используя завершение задержанной транзакции от цели, которая просто не может закончить транзакцию в определенное для запроса время. Поэтому мастер должен повторить любой запрос, который был закончен с типом "Retry", до тех пор, пока он не выполнится.

Вторая фаза. Цель автономно выполняет запрос на шине предназначения, используя зафиксированную информацию из задержанного запроса. Если задержанный запрос является запросом чтения, то цель извлекает требуемые данные и получает статус завершения. Если задержанный запрос является запросом записи, то цель записывает полученные от мастера данные и получает статус завершения. Результатом выполнения задержанного запроса на шине предназначения является *задержанное завершение*, которое состоит из зафиксированной информации задержанного запроса и статуса завершения (и данных при запросе чтения). Цель хранит задержанное завершение, пока мастер не повторит начальный запрос.

Третья фаза. Мастер повторно проходит алгоритм арбитража на шине и перевыдает первоначальный запрос. Цель дешифрирует запрос и передает мастеру статус завершения (и данные на запросе чтения). После этого задержанное завершение сбрасывается и транзакция выполнена. Статусом в данном случае является тип завершения транзакции, полученный целью после выполнения задержанного запроса (т. е. "Master-Abort", "Target-Abort", "ошибка четности", "нормально", "Disconnect" и т. д.) [9].

Информация задержанной транзакции

Алгоритм выполнения задержанной транзакции требует от цели фиксирования следующей информации:

- ❒ адреса;
- ❒ команды;
- ❒ сигнала разрешения обращения к байтам;
- ❒ четности адреса и данных, если установлен бит **Parity Error Response** (бит 6 регистра команд);
- ❒ REQ64# (для 64-разрядной передачи).

Выполнение транзакций записи требует фиксирования данных тех байт-линий, для которых выставлен соответствующий разряд сигнала разрешения

обращения к байтам. Цели не запрещается фиксировать данные байт-линий, для которых снят сигнал разрешения обращения к байтам. На транзакции чтения, адрес и команда доступны во время фазы адреса, а сигнал разрешения обращения к байтам доступен во время следующего такта. Этот сигнал действителен во время фазы данных для транзакций чтения и записи независимо от состояния линии IRDY#. На транзакции записи вся информация действительна в такое же время, как и на транзакции чтения, кроме фактических данных, которые являются действительными только тогда, когда выставлен сигнал IRDY#.

Данные для записи действительны, только когда выставлен сигнал IRDY#. Сигнал разрешения обращения к байтам действителен всегда для всей фазы данных, независимо от состояния линии IRDY#.

Цель отличает транзакции (тех же или различных мастеров) путем сравнения информации текущей транзакции с предварительно зафиксированной информацией (и для задержанного запроса (ов) и для задержанного завершения (й)). Во время транзакции чтения цель может не сравнивать сигнал разрешения обращения к байтам, если все байты возвращены независимо от состояния этого сигнала и адрес, по которому производится доступ, не имеет никаких побочных эффектов чтения (т. е. область, доступная для упреждающего чтения). Если информация текущей транзакции соответствует задержанному запросу (уже поставленному в очередь), цель не ставит запрос снова в очередь, а просто завершает транзакцию с типом завершения "Retry", указывая, что она еще не готова выполнить запрос. Если информация текущей транзакции соответствует задержанному завершению, то цель отвечает, сообщая статус и предоставляя данные, если данная транзакция является транзакцией чтения.

Мастер должен повторить транзакцию в точности как первоначальный запрос, включая данные для записи на всех байт-линиях (независимо от того, выставлен ли соответствующий байт разрешения или нет). В противном случае цель предположит, что это новая транзакция. Если первоначальная транзакция никогда не повторяется, то она будет, в конечном счете, отвергнута по истечении времени таймера. Два мастера теоретически могут запросить одинаковую транзакцию и цель не может и не должна отличить их и должна выполнить доступ. Если при генерации или выполнении задержанной транзакции происходит ошибка четности данных, применяются специальные правила [9].

Отмена задержанной транзакции

Устройству разрешено отказаться от задержанного запроса. Такое разрешение действует в интервале времени от момента, когда запрос поставлен в очередь, до тех пор, пока он не был предпринят на шине предназначения. Это

объясняется тем, что мастер обязан повторять запрос, пока он не выполнится. Как только запрос был предпринят на шине предназначения, он должен повторяться, пока не выполнится на шине предназначения и не может быть отменен. Мастеру разрешено представить другие запросы, но если мастер инициирует больше одного запроса, то он должен повторять все предпринятые запросы до тех пор, пока они все не выполнятся. Порядок повторения запросов строго не указан, но должен быть рационализирован. Когда задержанный запрос выполняется на шине предназначения, он становится *задержанным завершением*. Цели разрешено отменить задержанные завершения только в двух случаях.

- Первый случай — это когда задержанное завершение производит чтение из области, доступной для упреждающего чтения (команда *Memory Read Line* или *Memory Read Multiple*).
- Второй случай отмены применяется для всех задержанных завершений в том случае, если мастер не повторил запрос в пределах 2^{15} тактов. По истечении таймера устройство обязано отказываться от данных, иначе может произойти блокировка шины или устройства.

Отмена транзакции может привести к повреждению данных. Это происходит, когда отмененное задержанное завершение выполняет чтение из области, недоступной для упреждающего чтения. Если таймер отмены истекает, устройство может, но не должно сообщать об ошибке. Если данные являются предвыборкой (случай 1), рекомендуется, чтобы устройство не сообщало об ошибке. Однако если данные относительно доступа чтения не являются предвыборкой (случай 2), то рекомендуется, чтобы устройство сообщило об ошибке ее драйверу устройства [9].

Последствия использования задержанных транзакций

При выполнении задержанного запроса цель обязана выполнять все транзакции записи в память, адресованные к ней. Цель может, время от времени, повторять запись в память, пока решаются внутренние конфликты; например, когда переполнены все буферы данных; или перед тем, как задержанный запрос будет выполнен на шине предназначения (но будет гарантированно выполнен). Однако цель не может требовать выполнения задержанной транзакции на исходной шине перед получением данных для записи в память; иначе произойдет блокировка, как это описано далее.

В рассматриваемом примере блокировка происходит, когда мастер и цель находятся на различных шинах (или сегментах). Мост PCI-to-PCI[3], соеди-

[3] Такой мост мог быть разработан для устаревших версий спецификации PCI.

няющий эти две шины, не реализует задержанные транзакции. Мастер начинает запрос, который через мост отправляется к цели. Цель отвечает на запрос, используя задержанное завершение транзакции (законченную с типом "Retry"). Мост завершает запрос мастера с типом завершения "Retry" (не фиксируя запрос). Другой мастер (в том же сегменте шины, что и первоначальный мастер) собирается выполнить чтение из этого же устройства. Поскольку мост разработан для устаревшей версии спецификации, то он обязан передать данные для записи в память прежде, чем может быть повторена операция чтения. Если цель, которая использует задержанное завершение транзакции, не будет принимать данные для записи в память, до тех пор, пока мастер не повторит начальный запрос чтения, произойдет блокировка, потому что мост не может повторить запрос, пока цель не примет данные для записи. Для предотвращения подобной ситуации цель, использующая задержанное завершение транзакции, обязана принимать данные для записи в память даже при условии, что задержанная транзакция не завершена [9].

Поддержка многократных задержанных транзакций

В данном подразделе описывается поддержка многократных задержанных транзакций. В основе рассмотрения лежат основные правила для одной задержанной транзакции, описанные в предыдущем разделе. В качестве устройств, поддерживающих многократные задержанные транзакции, предполагаются мосты (в частности, мосты PCI-to-PCI). Использование мостов предназначается для повышения производительности системы и выполнения требования начального времени ожидания. Требования для поддержки многократных задержанных транзакций, описанные далее, относятся к мостам PCI-to-PCI. Такой подход позволяет пользоваться этой терминологией при описании транзакций, начатых на любом интерфейсе моста. Большинство других мостов (главный мост шины и стандартный мост шины расширения) будут обрабатывать только одну задержанную транзакцию. Для этих мостов также допускается поддержка многократных задержанных транзакций, но некоторые детали могут меняться. Фундаментальным требованием во всех случаях является соблюдение правил упорядочивания транзакций, описанных в *главе 5*. Соблюдение этих правил позволит избежать блокировок [9].

Транзакционные определения

- PMW (Posted Memory Write*)* — транзакция, которая выполнена на исходной шине до выполнения на шине предназначения и может происходить только для команд *Memory Write* и *Memory Write and Invalidate*.

- DRR (Delayed Read Request) — транзакция, которая должна выполниться на шине предназначения до выполнения на исходной шине и может быть инициирована командами: *Interrupt Acknowledge*, *I/O Read*, *Configuration Read*, *Memory Read*, *Memory Read Line* или *Memory Read Multiple*. Как упомянуто ранее, как только запрос был предпринят на шине предназначения, он должен продолжать повторяться, пока не выполнится на шине предназначения. До его выполнения DRR является только запросом и может быть отвергнут в любое время, в целях предотвращения блокировки или повышения производительности.
- DWR (Delayed Write Request) — транзакция, которая должна выполниться на шине предназначения до выполнения на исходной шине; может быть инициирована командой *I/O Write* или *Configuration Write*. Команды *Memory Write* и *Memory Write and Invalidate* должны быть буферизованы (PMW) и не должны выполняться как запрос DWR. Как упомянуто ранее, как только запрос был предпринят на шине предназначения, он должен продолжать повторяться, пока не выполнится. До его выполнения DWR является только запросом и может быть отвергнут в любое время.
- DRC (Delayed Read Completion) — транзакция, которая выполнена на шине предназначения и теперь передается на исходную шину для завершения. DRC содержит данные, которые требует мастер и статус завершения транзакции целью (нормальное завершение "Completion", завершение "Master-Abort", "Target-Abort", ошибка контроля четности и т. д.).
- DWC (Delayed Write Completion) — транзакция, которая выполнена на шине предназначения и теперь передается на исходную шину. DWC не содержит данные доступа, а только статус его завершения (нормальное завершение "Completion", завершение "Master-Abort", "Target-Abort", ошибка четности и т. д.). Данные для записи были записаны указанной цели [9].

Упорядочивающие правила для многократных задержанных транзакций

В табл. 4.1 представлены правила упорядочивания, когда мост в системе способен к обработке многократных транзакций, передаваемых в каждом направлении в одно и то же время. Число одновременных транзакций ограничено реализацией, а не архитектурой. Поскольку существует пять типов транзакций, которые могут быть обработаны в каждом направлении, следующая таблица имеет 25 ячеек. Из этих 25 ячеек в таблице только четыре указывают "No" (нет), восемь указывают "Yes" (да), остальные 13 указывают "No/Yes" (нет/да). Колонка таблицы представляет доступ, который был принят предварительно мостом, ряд представляет транзакцию, которая была принята сле-

дом за доступом, представленным в колонке. Таблица определяет отношения порядка между транзакциями, в котором они пересекают мост.

Таблица 4.1. *Упорядочивающие правила для многократных задержанных транзакций*

Запрос	PMW (2)	DRR (3)	DWR (4)	DRC (5)	DWC (6)
PMW (1)	No	Yes	Yes	Yes	Yes
DRR (2)	No	Yes/No	Yes/No	Yes/No	Yes/No
DWR (3)	No	Yes/No	Yes/No	Yes/No	Yes/No
DRC (4)	No	Yes	Yes	Yes/No	Yes/No
DWC (5)	Yes/No	Yes	Yes	Yes/No	Yes/No

No — указывает, что последующей транзакции не разрешено закончиться перед предыдущей транзакцией, чтобы сохранить порядок в системе. Четыре ячейки "No" представлены в колонке 2 и поддерживают последовательное представление данных в системе, как описано в модели "производитель — потребитель". Эти ячейки предотвращают передачу данных для записи доступа типа PMW.

Yes — четыре ячейки "Yes" в ряду 1 указывают, что PMW нужно разрешить быть выполненными прежде, чем задержанные запросы или задержанные завершения передадутся в том же направлении, в противном случае произойдет блокировка. PMW не может быть отсрочен от выполнения из-за того, что задержанный запрос или задержанное завершение были приняты раньше PMW. Единственное событие, которое может препятствовать выполнению PMW, это процесс получения доступа к шине или завершение целью транзакции с типом "Retry". Оба состояния являются временными и разрешаются независимо от других событий. Если мастер продолжает попытки выполнить задержанные запросы, он должен соблюдать рациональный порядок выполнения PMW. При выполнении последующей транзакции перед предшествующей нарушения порядка не происходит.

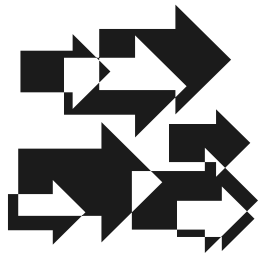
Четыре ячейки "Yes" в строках 4 и 5, колонки 3 и 4, указывают, что задержанные завершения должны пропустить задержанные запросы, передающиеся в том же направлении. Это предотвращает появление блокировок, когда два моста, поддерживающие задержанные транзакции, запрашивают доступы друг к другу. Если ни один из мостов не разрешит задержанным завершениям пропускать задержанные запросы, может возникнуть блокировка. Значение "Yes/No" указывает, что мост может в произвольном порядке разрешить выполнение последующей транзакции перед предыдущей. Реализация данной

таблицы разработчиком моста может влиять на стоимость самого моста и производительность системы [9].

Упорядочивание задержанных транзакций

Выполнение транзакции на исходной шине подразумевает получение в ответ на запрос мастера любого ответа, кроме "Retry". После этого события порядок задержанных транзакций считается установленным и не может быть изменен. Задержанные запросы и задержанные завершения происходят до выполнения транзакции на исходной шине и являются промежуточными этапами в процессе выполнения задержанной транзакции. Следовательно, для них необходимо установить порядок следования относительно друг друга. Допускается изменение порядка следования задержанных запросов относительно других задержанных запросов, задержанных запросов относительно задержанных завершений и задержанных завершений относительно других задержанных завершений. Не допускается изменение порядка следования относительно транзакций записи в память, которые указаны в табл. 4.1 (ячейки "No"). В общем случае мастер не должен ждать завершения запроса, чтобы инициировать другой запрос. В один момент времени мастер может иметь любое число запросов, законченных с типом завершения "Retry". Часть этих запросов может быть обработана как задержанные транзакции, а часть как обычные. Тем не менее при наличии нескольких незавершенных запросов в один момент времени мастер должен рационально повторять их для обеспечения возможности выполнения каждого. При некоторых условиях две транзакции мастера должны быть выполнены в определенном порядке. В этом случае мастер не должен инициировать их одновременно, а ждать завершения одной и только после этого начинать вторую. Данное правило выполняется для любого количества транзакций [9].

ГЛАВА 5



Порядок транзакций, арбитраж, оптимизация

Эффективное использование шины возможно при разработке правил распределения ресурсов между устройствами. Порядок получения доступа на шину определяется алгоритмом арбитража системы, от его реализации зависят многие параметры шины. Эти правила описываются в данной главе. Особое место занимает рассмотрение правил следования транзакций, в частности модель "производитель — потребитель". Методы, направленные на повышение производительности и оптимизацию работы шины, выделены в отдельный раздел, к ним относятся: использование сигнала разрешения обращения к байтам для указания значимых линий для передачи данных, оборотный цикл, комбинирование, объединение и свертка байт [9].

Правила следования транзакций

Для предотвращения коллизий, в терминах спецификации, называемых *тупиком* или *блокировкой*, необходимо определить порядок передачи, следования и завершения транзакций. Поэтому были разработаны правила упорядочивания транзакций на PCI, основными задачами которых является:

- упорядочивание результатов записи при использовании модели "производитель — потребитель". Это значит, записи производителя в область данных должны закончиться прежде, чем потребитель начнет читать эти данные. Данная модель разработана для описания правил обмена данными между двумя мастерами;
- буферизация транзакций с целью повышения производительности;
- предотвращение тупиковых состояний, или иными словами состояния блокировки шины, когда требуется очистить буфер для выполнения первого требования.

Порядок следования и передачи одной транзакции относительно других транзакций определен во время ее выполнения, т. е. после передачи данных. Транзакции, завершенные с типом завершения "Retry", не являются выполненными, поскольку передача данных не произошла. Следовательно, требования упорядочения относительно таких транзакций отсутствуют.

Транзакции, которые завершаются с типами завершения "Master-Abort" или "Target-Abort", рассматриваются как выполненные независимо от передачи данных и не будут повторены мастером. Система может принимать запросы в любом порядке, выполняя один при повторении другого. Если мастеру необходимо выполнить одну транзакцию перед выполнением другой, то он не должен инициировать вторую транзакцию, пока не выполнится первая. Транзакции могут быть разделены на две общие группы — буферизованные и небуферизованные — в зависимости от того, как они обрабатываются промежуточным агентом, например мостом.

Небуферизованные транзакции достигают цели их предназначения раньше, чем они завершаются в исходном устройстве. Мастер не может выполнять другие операции, пока транзакция не завершится в конечном адресате.

Один из методов повышения производительности — буферизация транзакций. Под *буферизацией* понимается сохранение транзакции промежуточным агентом во временном буфере. Основное назначение буферизации транзакций — развязать по времени двух агентов шины: того, который произвел транзакцию, и того, для кого она предназначена. Мастер не ждет, пока его транзакция достигнет адресата и имеет возможность выполнять другие операции, в том числе и транзакции шины. Таким образом, обеспечивается непрерывная работа мастера. При такой постановке задачи необходимо обеспечить аппаратную поддержку, что и реализовано в мостах PCI-to-PCI в виде буферов отложенной записи.

Буферизация ***разрешается*** для транзакций записи в память (команды *Memory Write* и *Memory Write and Invalidate*). Для транзакций чтения памяти (команды *Memory Read*, *Memory Read Line* и *Memory Read Multiple*), транзакций ввода-вывода (команды *I/O Read* и *I/O Write*) и транзакций конфигурации (команды *Configuration Read* и *Configuration Write*) буферизация ***запрещена*** (исключение составляют главные мосты). В сущности, ответственность за то, что транзакция достигнет конечного адресата, несет промежуточный агент доступа (например, мост). Существуют две категории устройств с различными требованиями упорядочивания и буферизации транзакций. Каждая категория будет рассмотрена отдельно [9].

Очередь транзакций для простых устройств

Под простым устройством понимается любое устройство, которое при действии как мастер шины не требует, чтобы его данные для записи были буфери-

зованы в шинной логике интерфейса. Вообще устройства, которые не соединяются с локальными или центральными процессорами, считаются простыми устройствами. Конечные автоматы цели и мастера в интерфейсе PCI простого устройства полностью независимы. Устройство не может выполнить любую транзакцию (буферизованную или небуферизованную) как цель, зависящая от предыдущего завершения любой другой транзакции как мастер. Простым устройствам разрешено завершить транзакцию с типом завершения "Retry" для выполнения транзакции как задержанной. Также допускается завершение транзакции с типом "Retry" при наличии внешних ограничений, например, во время регенерации видеоизображения или при заполнении буфера транзакциями, передаваемыми в одном и том же направлении [9].

Интерфейсная зависимость "мастер — цель"

Рассмотрим пример тупикового состояния, в которое может перейти шина, если цель и мастер устройства интерфейсно-зависимы. Предположим, что два устройства, устройство А и устройство В, взаимодействуют непосредственно друг с другом. Оба устройства одновременно предпринимают попытки записи в пространство ввода-вывода. Предположим, что устройству А первому предоставляется шина и оно выполняет запись в пространство ввода-вывода, обращаясь к устройству В. Устройство В дешифрирует адрес и выставляет сигнал DEVSEL#. Далее предположим, что устройство В нарушает требование независимости конечного автомата цели от конечного автомата мастера и всегда завершает транзакции как цель с типом завершения "Retry", пока конечный автомат мастера не выполнит его запросы. Поскольку устройство В также должно выполнить транзакцию ввода-вывода как мастер, оно завершает транзакцию устройства А с типом "Retry". Затем устройству В предоставляется шина, и устройство В выполняет транзакцию записи в пространство ввода-вывода, обращаясь к устройству А. Если устройство А отвечает тем же способом, что и устройство В, система блокирована бесконечными запросами "Retry" [9].

Отмена буферизованных данных

Система может переходить в тупиковое состояние, когда устройство не принимает транзакцию записи в память от моста. Как описано далее, мост должен в некоторых случаях очистить его буфер как мастер прежде, чем он завершит транзакцию как цель. Предположим, что мост PCI-to-PCI содержит буферизованные данные для записи в память, адресованные к нижерасположенному устройству. Но прежде, чем мост может получить доступ к нижерасположенной шине, чтобы произвести транзакцию записи, устройство на этой шине начинает чтение из памяти. Поскольку требование 3 в правилах моста указывает, что буферы должны быть очищены прежде, чем может быть

закончена транзакция чтения, мост должен выполнить повторное чтение от агента и предпринять попытку транзакции записи. Если нижерасположенное устройство должно было принять данные для записи, зависящие от предыдущего завершения повторенной транзакции чтения (т. е. если оно не могло бы принять буферизованную запись, пока сначала не завершило бы транзакцию чтения), шина будет блокирована. Некоторые мосты PCI-to-PCI, разработанные для устаревших версий спецификации PCI, требуют, чтобы их буфер был очищен перед началом любой небуферизованной транзакции. Следовательно, возможно состояние блокировки, если нижерасположенное устройство принимает буферизованную запись в зависимости от предшествующего завершения любой небуферизованной транзакции.

Требуемая независимость целевых и главных конечных автоматов в простом устройстве подразумевает, что простое устройство не может внутренне буферизовать никаких исходящих транзакций. Например, если во время выполнения его основной функции устройство должно выполнить запись в память как мастер на шине PCI, устройство не может буферизовать эту запись памяти в интерфейс мастера в устройстве. Более точно, устройство не может выполнять внутренние операции типа модификации регистров состояния, которые могут быть обнаружены другим мастером в системе. Простое устройство должно ожидать, пока транзакция записи памяти выполняется на шине (выставлен сигнал TRDY#; завершение типа "Master-Abort" или "Target-Abort") перед внутренними операциями. Простым устройствам рекомендуется буферизовать входящие транзакции записи в память для ускорения транзакций на шине. Упорядочивание входных буферизованных данных для записи зависит от конкретной реализации. Простые устройства не поддерживают монопольные доступы и не должны использовать сигнал LOCK# [9].

Правила следования транзакций для мостов

С точки зрения выполняемых действий устройство типа "мост" — это любое устройство, которое осуществляет внутреннюю буферизацию исходящих транзакций записи в память. То есть транзакций записи, которые должны быть выполнены устройством как мастером на шине PCI. Мосты обычно соединяют две шины типа PCI, главную шину и PCI-шину, или PCI-шину и шину для локального процессора.

Мостам разрешено буферизовать транзакции записи в память, передающиеся в любом направлении через мост. Следующие правила упорядочения гарантируют, что мастера в системе получают информацию о результатах выполнения транзакций записи в надлежащем порядке, даже при условии буферизации транзакции записи мостом. Они также гарантируют, что шина не блокируется, когда мост очищает его буферы.

- Буферизованная запись в память передается в таком же направлении через мост и завершится на адресуемой шине в таком же порядке, в котором она выполняется на исходной шине. В результате завершения с типом "Disconnect" один блок данных исходной шины может быть разбит на несколько транзакций на адресуемой. Такие транзакции не должны допускать изменения порядка завершения фаз данных на адресуемой шине относительно исходной.
- Транзакции записи, передаваемые в одном направлении через мост, не имеют никаких требований упорядочивания относительно записи, передаваемой в другом направлении.
- Буферы отложенной записи для обоих направлений должны быть очищены перед выполнением транзакции чтения, передаваемой в любом направлении. Буферизованная запись в память, возникшая на одной стороне моста с транзакцией чтения и выполняемая на исходной шине раньше команды чтения, должна завершиться на адресуемой шине в таком же порядке. Пусть буферизованная запись в память и транзакция чтения находятся по разные стороны моста, исходная шина записи в память является шиной предназначения для транзакции чтения и наоборот. Запись в память выполняется на своей исходной шине раньше выполнения команды чтения на шине предназначения (транзакции чтения). В этом случае запись должна быть выполнена на шине ее предназначения в таком же порядке относительно транзакции чтения. Другими словами, транзакция чтения должна пропустить любую буферизованную запись через мост, возникшую на этой же стороне и буферизованную раньше чтения. Кроме того, прежде чем транзакция чтения может выполниться на исходной шине, она должна "вытолкнуть" из моста любую буферизованную запись, которая возникла на противоположной стороне и была буферизована до выполнения команды чтения на шине предназначения.
- Мост ни при каких условиях не должен принимать транзакции записи в память в качестве цели, зависящей от предыдущего завершения неблокированной транзакции в качестве мастера на той же самой шине. Мост может принимать транзакции записи в память в качестве цели, зависящей от предыдущего завершения блокированной транзакции в качестве мастера, при условии, что мост уже установил монопольный доступ к цели предназначения данной транзакции; в противном случае может произойти блокировка. Во всех других случаях мостам разрешено отказаться принимать запись в память только при условии неразрешимых событий, например, при заполнении буфера памяти предыдущими транзакциями записи в память, передаваемых в одном направлении.

Главным мостам разрешено буферизовать транзакции записи ввода-вывода, которые возникают на главной шине и заканчиваются в сегменте PCI. При

этом должны выполняться упорядочивающие правила PCI. Таким образом, буферизация мостом транзакции записи ввода-вывода, возникшей на главной шине, и ее выполнение на шине PCI не должно приводить к блокировкам. Транзакция выполнится на адресуемой шине PCI перед завершением на исходной шине. Поскольку транзакции записи памяти могут быть буферизованы мостами, а запись ввода-вывода может быть буферизована главным мостом, то мастер не располагает информацией о времени выполнения транзакции в конечной шине предназначения. Драйвер устройства должен гарантировать, что запись выполнена по указанному адресу (а не в промежуточный мост). Для этого драйвер должен произвести чтение из устройства, которое было целью записи. Чтение из памяти или из пространства ввода-вывода вынуждает мосты, располагающиеся между мастером и целью, очищать все буферизованные данные до выполнения чтения. Запросы на прерывание, использующие линии INTx#, не появляются на шине PCI в качестве транзакций, поэтому на них не распространяются правила упорядочивания транзакций. Кроме того, от системы не требуется использования транзакции шины "Interrupt Acknowledge" для обработки прерывания. Таким образом, прерывания не являются синхронизируемыми событиями, и очищение буферов драйверами устройств не может зависеть от прерываний. Правила упорядочивания для транзакций записи в память применяются к транзакциям MSI *(Message Signaled Interrupt)* [9].

Модель "производитель — потребитель"

Взаимодействие двух мастеров на шине описывается моделью "производитель — потребитель". Модель состоит из пяти взаимосвязанных элементов: производитель, потребитель, данные, флаг и элемент статуса. Производитель создает данные, а потребитель их потребляет. Взаимодействие между производителем и потребителем осуществляется посредством флага и элемента статуса. Производитель устанавливает флаг, тем самым указывая, что все данные записаны и их можно забирать. Затем он ожидает, когда потребитель получит данные и вернет код статуса завершения. Потребитель дожидается установления флага, затем он его сбрасывает, забирает данные, и возвращает код статуса завершения. Производитель получает код статуса завершения, очищает его, и последовательность действий повторяется. Очевидно, важную роль играет порядок, в котором записываются флаг и данные. Если часть данных, записанных производителем, была сохранена в буфере, то без правил очищения/сброса буфера потребитель мог бы получить неверную информацию: установку флага прежде, чем завершится запись данных. Упорядочивающие правила PCI разработаны таким образом, что независимо от того, какая часть записи была буферизована, потребитель никогда не сможет получить флаг и читать данные, пока не закончится запись данных. В специфика-

ции PCI данное условие называется как "наличие последовательного представления данных". Еще одним условием является то, что, если в дополнение к коду статуса потребитель должен был передать информацию производителю, становится важными порядок записи этой дополнительной информации и кода статуса, как это описано для данных и флага.

На практике флагом может быть регистр типа "дверного звонка" (doorbell) в устройстве, или он мог бы быть указателем главной памяти на данные, расположенные в какой-либо области памяти. Потребитель мог бы передавать сообщения производителю, используя прерывание или другой регистр doorbell, вместо того, чтобы опрашивать элемент статуса производителя. Механизм использования регистра "дверного звонка" (doorbell) состоит в следующем: он предназначен для обмена сообщениями и должен быть обработан, как только изменилось его состояние. Для того чтобы послать сообщение, производитель записывает в адресном пространстве шины PCI полезные данные, а затем обращается ("звонит") в 64-разрядный регистр doorbell потребителя. После этого записанное сообщение ставится в очередь к регистру doorbell. Потребитель обрабатывает последовательные doorbell в порядке их поступления.

Во всех случаях основной смысл остается одинаковый: записи производителя в области данных должны закончиться прежде, чем потребитель прочитает состояние флага и начнет читать данные.

Данная модель обеспечивает постоянное нахождение в системе всех ее элементов: данных, флага, элемента статуса, производителя и потребителя. Каждый из них может находиться на различных шинах, и упорядочивающие правила поддерживают последовательное представление данных. Рассмотрим пример, представленный на рис. 5.1. Агент, формирующий данные, флаг и элемент статуса находятся на шине 1, фактические данные и потребитель находятся на шине 0. Производитель записывает последние данные, и мост PCI-to-PCI между шинами 0 и 1 выполняет доступ, сохраняя данные в буфере. Тогда производитель данных записывает флаг, изменяющий его статус, чтобы указать, что данные теперь готовы для использования потребителем. В этом случае флаг был установлен до того, как в конечный адрес была фактически записана заключительная данная величина. Упорядочивающие правила PCI требуют, чтобы чтение потребителем флага заставляло мост PCI-to-PCI сбросить буферизованные данные для записи в конечный адрес до завершения чтения. Потребитель, проверяя флаг, определяет, что данные готовы; в это время фактически данные уже находятся по конечному адресу.

Упорядочивающие правила работают независимо от того, где фактически размещены производитель, потребитель, данные, флаг и элемент статуса. Данные всегда находятся в месте конечного предназначения прежде, чем по-

требитель может прочитать флаг. Это правило выполняется, даже когда все пять составляющих находятся на различных сегментах шины системы. В одном варианте конфигурации данные будут переданы адресату при чтении потребителем флага. В другом варианте конфигурации чтение флага происходит без передачи данных адресату; однако запрос чтения фактических данных выталкивает (помещает) заключительную величину по адресу предназначения перед выполнением чтения.

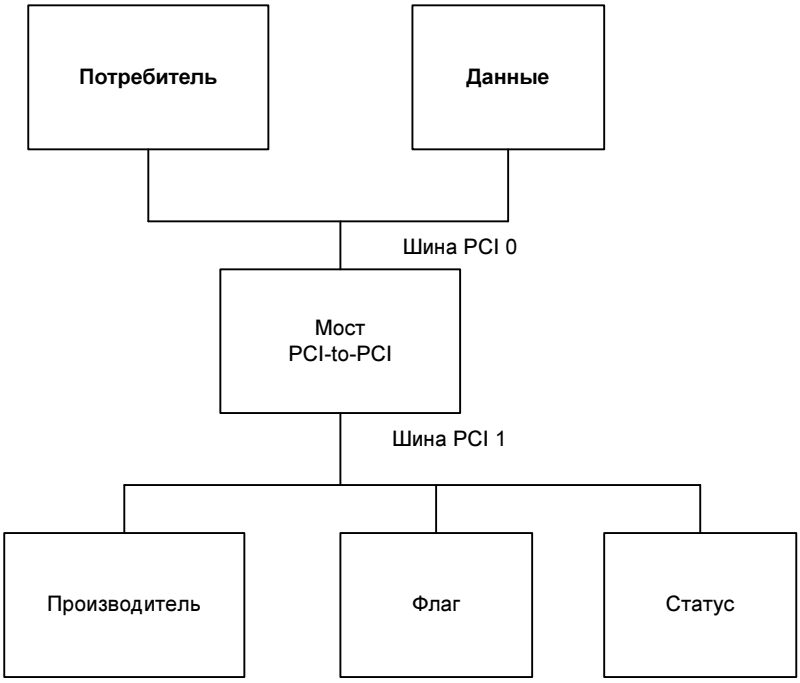


Рис. 5.1. Пример модели "производитель — потребитель"

Система может иметь много пар "производитель — потребитель", работающих одновременно, с различными множествами "данные — флаг — статус", разбросанными внутри системы. Но поскольку только один производитель может записать единственный набор "данные — флаг", нет никаких требований упорядочивания между различными мастерами. Запись мастера на шине может происходить в одном порядке на шине по отношению к записи другого мастера и в другом порядке на другой шине. В этом случае, правила допускают перестроение некоторых записей; например, агент на шине 1 может "видеть" завершение транзакции A от мастера на шине 1 первым, после транзакции B от другого мастера на шине 0. Агент на шине 0 может "видеть" завершение транзакции B первым, после транзакции A. Даже при том, что фак-

тически транзакции завершаются в различном порядке, это не вызывает сбоев, т. к. различные мастера должны обращаться к разным наборам "данные — флаг" [9].

Перечень упорядочивающих требований PCI

Описанные правила необходимо свести в итоговый перечень. Общие требования относятся ко всем транзакциям PCI, независимо от того, передаются ли они в виде задержанных транзакций или в качестве обычных. Требования упорядочивания задержанных транзакций приведены отдельно.

Общие требования

1. Порядок транзакции определен при ее выполнении. Транзакции, законченные с типом "Retry", являются только лишь запросами и могут быть обработаны системой в любом порядке.
2. Записи в память могут быть сохранены в буфере моста в обоих направлениях. Записи в пространство ввода-вывода и в конфигурационное пространство не могут быть сохранены в буфере. (Запись ввода-вывода может быть сохранена в буфере главного моста, но с некоторыми ограничениями.) Транзакции чтения (пространства памяти, ввода-вывода или конфигурации) не сохраняются в буфере.
3. Буферизованные записи в память, передаваемые в одинаковом направлении через мост, выполнятся на шине предназначения в таком же порядке, как и на исходной.
4. Транзакции записи, пересекающие мост в противоположных направлениях, не имеют никаких упорядочивающих отношений.
5. Транзакция чтения должна пропустить вперед через мост любую буферизованную запись, возникшую на *той же* стороне моста и буферизованную *раньше* операцию чтения. Прежде чем транзакция чтения может выполниться на исходной шине, она должна "вытолкнуть" из моста любую буферизованную запись, которая возникла на *противоположной* стороне и была буферизована *до* выполнения команды чтения на шине предназначения.
6. Мост ни при каких условиях не должен принимать транзакции записи в память в качестве цели, зависящей от предыдущего завершения неблокированной транзакции в качестве мастера на той же самой шине. Мост может принимать транзакции записи в память в качестве цели, зависящей от предыдущего завершения мостом блокированной транзакции в качестве мастера, при условии, что мост уже установил монопольный доступ к цели предназначения данной транзакции; в противном случае может произойти

блокировка. Во всех других случаях, мостам разрешено отказаться принимать запись в память только при условии неразрешимых событий, например, при заполнении буфера памяти предыдущими транзакциями записи в память, передаваемых в одном направлении [9].

Перечень упорядочивающих требований задержанных транзакций

1. Цель, которая поддерживает задержанные транзакции, при соответствующей разработке может иметь любое количество одновременно инициированных задержанных транзакций.
2. Только небуферизованные транзакции могут быть обработаны как задержанные.
3. Мастер должен повторить любую транзакцию, завершенную с типом "Retry", т. к. данная транзакция может быть задержанной.
4. Запрос, инициированный на шине предназначения, должен повторяться, пока не выполнится. До его возникновения запрос может быть отменен в любое время.
5. Задержанное выполнение может быть отменено в двух случаях. В первом случае отмена возможна при обращении в область, доступную для упреждающего чтения. Во втором случае отмена допускается, если мастер не повторил транзакцию через 2^{15} тактов.
6. Цель должна принимать все записи в память, адресованные к ней, даже при выполнении запроса, использующего задержанное выполнение транзакции.
7. Задержанные запросы и задержанные выполнения не требуют упорядочивания относительно друг друга.
8. Только задержанное выполнение записи (DWC) может пропустить буферизованную запись в память (PMW). Буферизованная запись в память должна пропустить все транзакции, кроме другой буферизованной записи в память.
9. Один мастер может инициировать любое число исходящих запросов, завершенных с типом "Retry". Если существует необходимость выполнения одной транзакции перед другой, то мастер не должен начинать вторую транзакцию до выполнения первой.

Правила следования запросов

Запросом называется транзакция, представленная на шине. Транзакция, завершенная с типом завершения "Retry", считается запросом. Транзакция ста-

новится *законченной* или *завершенной*, когда данные фактически переданы (или в случае завершения способом "Master-Abort" или "Target-Abort"). Дальнейшее рассмотрение будет относиться к транзакциям, как к запросам или выполнениям в зависимости от результата их завершения.

На транзакцию, законченную с типом "Retry", не распространяются упорядочивающие правила. Порядок доступов определен только во время выполнения доступа (во время передачи данных). Например, четыре мастера A, B, C, и D находятся в одном шинном сегменте и всем требуется доступ на шину. В этом примере каждый агент может запросить только одну транзакцию в один момент времени и не будет запрашивать другой, пока не завершится текущий доступ. Порядок, в котором завершаются транзакции, основан на алгоритме арбитража и ответа цели, а не на порядке выставления сигнала REQ# каждого агента. Учитывая, что некоторые запросы завершены с типом "Retry", порядок, в котором они выполняются, не зависит от порядка, в котором они были первоначально запрошены. Изменением алгоритма арбитража выполнение транзакций может проходить в любой последовательности (т. е. A, B, C и затем D или B, D, C и затем A и т. д.). Поскольку арбитр может изменить порядок, в котором транзакции запрошены на шине, и, следовательно, порядок выполнения этих транзакций, то системе разрешается выполнять их в любом порядке. Это означает, что запрос от любого агента не имеет отношений порядка следования относительно запроса любого другого агента. Единственное исключение из этого правила составляет использование сигнала LOCK#.

Пусть четыре мастера (A, B, C и D) находятся в одной микросхеме (многофункциональное устройство). Для многофункционального устройства эти четыре мастера работают независимо друг от друга, и каждая функция представляет только единственный запрос на шине. Порядок выполнения их запросов является таким же, как если бы они были отдельными агентами. Поэтому запросы от одного агента могут выполниться в любом порядке, т. к. они не имеют никаких отношений друг с другом.

Рассмотрим другое, не многофункциональное, устройство, которое имеет много внутренних ресурсов. Эти ресурсы могут инициировать транзакции на шине и имеют некоторые отношения порядка, т. е. зависят друг от друга. В таком случае устройство должно гарантировать, что в один момент времени на шине представлен только один запрос. Агент не должен инициировать следующую транзакцию до завершения предыдущей. Например, устройство имеет две транзакции на шине: транзакцию A и транзакцию B. Транзакция A должна выполниться раньше транзакции B, чтобы удовлетворить требованиям внутреннего порядка. В этом случае мастер не может инициировать транзакцию B до выполнения транзакции A. Рассмотрим пример нарушения правил внутреннего порядка и последствия этого нарушения. Пусть целью тран-

закции A является флаг, указывающий состояние данных, а целью транзакции B — сами данные. Транзакция A завершается с типом "Retry", потому что цель, к которой она обращается, в настоящее время занята или находится с другой стороны моста. Транзакция B направлена к цели, которая готова к взаимодействию и выполнит запрос немедленно. Если мастер разрешит транзакции B быть представленной на шине после того, как транзакция A была закончена с типом "Retry", транзакция B может выполниться раньше транзакции A. В этом случае происходит доступ к данным до их фактического наличия. Данную ситуацию должен был предотвратить мастер. Для этого он не должен был инициировать транзакцию B после выполнения транзакции A. Мастер, который инициирует несколько запросов на шине, должен гарантировать, что последовательные запросы (которые имеют какое-либо отношение к предыдущему запросу), не будут представлены на шине до выполнения предыдущего запроса. Система позволяет выполнить запросы от одного агента в любом порядке. Если мастер допускает инициирование запросов без их завершения, то он должен повторить каждый запрос независимо от других запросов [9].

Упорядочивание задержанных транзакций

Процесс выполнения задержанной транзакции состоит из 3 этапов:

1. Запрос мастера.
2. Выполнение запроса целью.
3. Выполнение транзакции мастером.

Во время первой фазы мастер генерирует транзакцию на шине, цель дешифрирует доступ, фиксирует информацию, требуемую для завершения доступа, и заканчивает запрос с типом завершения "Retry". Зафиксированная информация запроса называется *задержанный запрос*. Во время второй стадии цель независимо выполняет запрос на шине предназначения, используя зафиксированную информацию задержанного запроса. Результатом выполнения задержанного запроса на шине предназначения является *задержанное выполнение*, которое состоит из зафиксированной информации задержанного запроса и статуса завершения (и данных для запроса чтения). Во время третьей стадии мастер проходит алгоритм арбитража на шине и перевыдает первоначальный запрос. Цель дешифрирует запрос и возвращает мастеру статус завершения (и данные для запроса чтения). После этого удаляется задержанное выполнение и транзакция выполнена.

Число одновременно обрабатываемых мостом задержанных транзакций зависит от реализации, а не от архитектуры. Табл. 5.1 представляет правила упорядочивания для системы, в которой мост способен обеспечить передачу не-

скольких транзакций в каждом направлении в одно и то же время. Каждый столбец таблицы представляет доступ, первоначально предпринятый мостом, каждая строка представляет транзакцию, предпринятую после доступа. Содержание ячейки указывает порядок следования второй транзакции по отношению к первой. Кроме того, в таблице используются следующие сокращения транзакций, определение которых приведено в *главе 4*, и обозначения:

- **PMW** (Posted Memory Write);
- **DRR** (Delayed Read Request);
- **DWR** (Delayed Write Request);
- **DRC** (Delayed Read Completion);
- **DWC** (Delayed Write Completion);
- **No** — указывает, что последующей транзакции не позволено закончиться перед предыдущей для выполнения порядка следования. Четыре ячейки "No" представлены в столбце 2 и поддерживают последовательное представление данных в системе, как описано в модели "производитель — потребитель". За счет этих правил предотвращается передача данных для записи доступа типа PMW;
- **Yes** — указывает, что последующей транзакции нужно позволить выполниться до предыдущей, в противном случае может произойти блокировка.

  При возникновении блокировки транзакция PMW должна пропустить задержанную транзакцию. Если мастер продолжает попытки выполнить задержанные запросы, он должен в наиболее рациональном порядке выполнять PMW. Нарушения порядка не происходит, когда последующая транзакция выполняется перед предшествующей;
- **Yes/No** — данное значение указывает, что мост может выполнять транзакции в произвольном порядке. Реализация данной таблицы разработчиком моста может влиять на стоимость самого моста и производительность системы.

Таблица 5.1. *Правила следования для мостов*

Запрос строки пропускает запрос столбца?	PMW (2)	DRR (3)	DWR (4)	DRC (5)	DWC (6)
PMW (1)	No[1]	Yes[5]	Yes[5]	Yes[7]	Yes[7]
DRR (2)	No[2]	Yes/No	Yes/No	Yes/No	Yes/No
DWR (3)	No[3]	Yes/No	Yes/No	Yes/No	Yes/No
DRC (4)	No[4]	Yes[6]	Yes[6]	Yes/No	Yes/No
DWC (5)	Yes/No	Yes[6]	Yes[6]	Yes/No	Yes/No

Правило 1. Вторая транзакция PMW не может быть передана раньше первой PMW (столб. 2, стр. 1).

Буферизованные транзакции записи в память должны выполняться в том порядке, в котором они предприняты. Если вторая запись осуществляется во флаг данных, то потребитель может получить неверные данные.

Правило 2. Транзакция чтения должна пропустить буферизованные данные для записи при выполнении правил упорядочивания (столб. 2, стр. 2).

Например, запись и чтение производится в одно и то же местоположение памяти. В этом случае если чтение произойдет раньше записи, то оно не получит данных для записи, которые для него и предназначались. По этим же соображениям чтение ввода-вывода не может опередить PMW.

Правило 3. Небуферизованная транзакция записи должна "вытолкнуть" буферизованные данные для записи, чтобы сохранить порядок (столб. 2, стр. 3).

Задержанный запрос записи может быть флагом предварительно записанных (PMW) данных, и поэтому флаг записи не может быть передан раньше данных, которые он потенциально охватывает.

Правило 4. Транзакция чтения должна "выталкивать" данные записи обратно к исходной шине транзакции чтения (столб. 2, стр. 4).

Например, чтение из регистра статуса устройства, пишущего данные в память, не должно выполниться прежде, чем данные перемещены обратно на шину происхождения; иначе могут использоваться неверные данные.

Правило 5. Буферизованная запись в память должна выполниться раньше задержанного запроса (чтения или записи) с целью предотвращения блокировок (столб. 3 и 4, стр. 1).

Блокировка может произойти при совместном использовании мостов, которые поддерживают задержанные транзакции, с мостами, неподдерживающими задержанные транзакции. Рассмотрим рис. 5.2. Блокировка может произойти при наличии следующих условий. Мосты X и Z не используют задержанные транзакции. Мост Y использует задержанные транзакции, мост Y находится между мостами X и Z. Мастер 1 производит чтение из цели 1, которое передано через мост X и поставлено в очередь как задержанный запрос мостом Y. Мастер 3 осуществляет чтение из цели 3, которое передано через мост Z и поставлено в очередь как задержанный запрос мостом Y. После того как мастера 1 и 3 завершат доступ с типом "Retry", мастера 2 и 4 начинают транзакции записи в память большой продолжительности, обращаясь к целям 2 и 4 соответственно, которые сохраняются в буферах записи мостов X и Z соответственно. Когда мост Y пытается выполнить чтение в любом направлении, мосты X и Z должны очистить их буферы записи перед тем, как разрешить запросу чтения пройти через них.

Если буферы записи мостов X и Z большего размера, чем такие же буферы моста Y, то буферы моста Y заполнятся. Если буферизованным данным записи не разрешено пропустить DRR, система перейдет в заблокированное состояние, т. е. "зависнет". Мост Y не может отказаться от запроса чтения, т. к. он был предпринят, и он не может принять больше данных записи, пока не закончится чтение в противоположном направлении. Поскольку это условие существует для обоих направлений, то ни один DRR не может выполниться, потому что другой блокирует путь. Поэтому данные PMW должны передаваться в транзакции DRR, пока DRR блокирует прямое продвижение данных в транзакции PMW.

То же самое условие существует, когда DWR находится во главе обеих очередей, т. к. некоторые мосты также требуют, чтобы буферы отложенной записи были очищены на небуферизованном цикле записи.

Правило 6. Задержанное выполнение (чтения или записи) должно быть выполнено до задержанного запроса (чтения или записи) с целью предотвращения блокировок (столб. 3 и 4, стр. 4 и 5).

Блокировка может произойти, когда два моста, которые поддерживают задержанные транзакции, запрашивают доступы друг к другу. Общий сегмент шины PCI находится на вторичной шине по отношению к мосту A и на первичной шине по отношению к мосту B. Если ни один мост не разрешает задержанным выполнениям пропускать задержанные запросы, то система "зависнет".

Например, пусть запрос моста A к мосту B выполняется на вторичной шине моста B, и запрос моста B заканчивается на первичной шине моста A. Выполнение моста A теперь находится в очереди после запроса моста B и выполнение моста B в очереди после запроса моста A. Если ни один из мостов не разрешает выполнениям пропускать запросы, то произойдет блокировка.

Правило 7. Буферизованная запись в память должна пропустить задержанное выполнение (чтения или записи) с целью предотвращения блокировок (столб. 5 и 6, стр. 1).

Как в примере приведенного при описании правила 5, для конфигурации, представленной на рис. 5.2, может произойти другой вариант блокировки. Пусть DRC находится во главе очередей в обоих направлениях моста Y в одно и то же время. Мосты (X и Z) содержат буферизованные данные записи от другого мастера. В этом случае проблема заключается в том, что транзакция чтения не может быть повторена, пока все буферизованные данные для записи не будут переданы из моста, и мастеру разрешено повторить его первоначальный запрос. В итоге мост не сможет принять больше буферизованных данных, потому что его внутренние буферы заполнены, и он не может очистить их, пока не выполнится DRC. Если такое состояние существует в обоих

направлениях, ни один DRC не может выполниться, потому заблокирован тракт передачи данных. Поэтому данные PMW должны пропускать DRC, если DRC блокирует PMW. Данное правило выполняется для случая, когда во главе обеих очередей находится DWC [9].

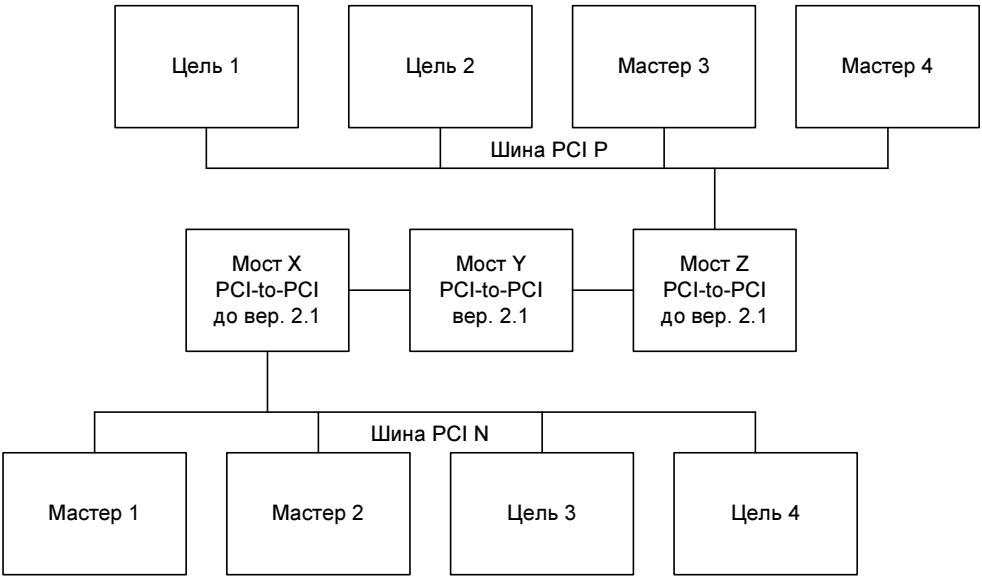


Рис. 5.2. Пример системы с мостами PCI-to-PCI

Независимость транзакций

Некоторые транзакции, поставленные в очередь как задержанные запросы или задержанные выполнения, не имеют никаких отношений порядка с любыми другими задержанными запросами или задержанными выполнениями. С целью повышения производительности или снижения стоимости допускается устанавливать порядок следования задержанных запросов и задержанных выполнений.

- Задержанные запросы могут пропускать другие задержанные запросы (столб. 3 и 4, стр. 2 и 3). Поскольку задержанные запросы не имеют никаких отношений порядка с другими задержанными запросами, то значения этих ячеек не играют роли.
- Задержанные запросы могут пропускать задержанные завершения (столб. 5 и 6, стр. 2 и 3). Поскольку задержанные запросы не имеют никаких отношений порядка с задержанными выполнениями, то значения этих ячеек не играют роли.

- Задержанные выполнения могут пропускать другие задержанные выполнения (столб. 5 и 6, стр. 4 и 5). Так как задержанные выполнения не имеют никаких отношений порядка с другими задержанными выполнениями, то значения этих ячеек не играют роли.
- Задержанные выполнения записи могут пропускать буферизованную запись в память или быть блокированными ими (столб. 2, стр. 5).

Если DWC разрешено пропустить PMW или если порядок их следования не меняется, то при любых условиях отсутствует блокировка или несогласованность данных. Данные транзакции DWC и данные транзакции PMW передаются в противоположных направлениях. Обе транзакции начаты мастерами, находящимися на различных шинах и обращающихся к целям на различных шинах [9].

Задержанные транзакции. Применение сигнала LOCK#

Мост должен поддерживать протокол использования сигнала LOCK#, если транзакция начата на его основной шине (и используется протокол блокировки), но не обязан поддерживать LOCK# для транзакций, начатых на его вторичной шине. Если блокированная транзакция начата на основной шине, и мост является целью, то мост должен выполнять правила блокированного доступа. Мост обязан выполнить ("вытолкнуть") все PMW, предпринятые со стороны основной шины, во вторичную шину до начала монопольного доступа на вторичной шине. Мост может отменить любые поставленные в очередь запросы, разрешить блокированной транзакции пропускать поставленные в очередь запросы, или просто выполнять все находящиеся в очереди транзакции перед выполнением монопольного доступа на вторичном интерфейсе. Как только блокированная транзакция была поставлена в очередь мостом, мост не может принимать другие транзакции от основного интерфейса до окончания монопольного доступа, исключая продолжение доступа непосредственно мастером. Пока монопольный доступ не установлен на вторичном интерфейсе, мосту разрешено ставить в очередь транзакции от вторичного интерфейса, но не от основного. Как только на вторичном интерфейсе установлен монопольный доступ, мост не может принимать буферизованные данные записи, передаваемые в основной интерфейс до освобождения сигнала LOCK# (сигналы FRAME# и LOCK# сняты на одном такте). В наиболее простой реализации мост не принимает транзакции ни в одном из направлений, как только на вторичной шине установлен монопольный доступ. Мост должен выполнить транзакции PMW, DRC, и DWC, передаваемые на основную шину до выполнения монопольного доступа на его исходной шине. Перечисленные правила достаточны для функционирования системы без создания тупиковых состояний. Допускается урезанная реализация моста или уст-

ройства, но во всех случаях должно гарантироваться функционирование без состояний блокировки шины [9].

Условия возникновения ошибок

Мост может отменить данные или статус транзакции, которая была выполнена, используя задержанное выполнение транзакции, если мастер не повторил запрос в пределах 2^{10} тактов на шине PCI (приблизительно 30 мкс на 33 МГц). Однако рекомендуется, чтобы мост не отменял транзакцию до 2^{15} тактов PCI (приблизительно 983 мкс на 33 МГц) после того, как он получил данные или статус. Малое значение применимо, если мастер разработан для устаревшей версии спецификации PCI и не в состоянии повторить транзакцию в точности. В этом случае мост может быть запрограммирован, чтобы отменить задержанное завершение раньше и разрешить выполнение других транзакций. В обычном случае мост ожидает более длительное время, если повторение транзакции задерживается другим мостом или мост разработан для устаревшей версии спецификации, которая не поддерживала задержанные транзакции.

Когда истекает время таймера (называемого *таймером отмены*), устройство обязано отказываться от данных; в противном случае может произойти блокировка.

Отмена транзакции может привести к повреждению данных. Это происходит, когда отмененное задержанное выполнение производит чтение из области, не доступной для упреждающего чтения.

По истечении таймера устройство может сообщить об ошибке. Данное требование необязательно для выполнения и носит рекомендательный характер.

Если данные доступны для упреждающего чтения, то рекомендуется, чтобы устройство не сообщало об ошибке, т. к. это не может привести к повреждению данных. Рекомендуется, чтобы устройство сообщало об ошибке ее драйверу, если данные находятся в области, не доступной для упреждающего чтения. Мост может сообщать об ошибке установкой сигнала SERR#, т. к. обычно для моста не требуются драйверы устройства [9].

Арбитраж

Арбитраж, или судейство шины, является неотъемлемым механизмом для обеспечения стабильной работы шины, распределения ресурсов агентам шины, и следовательно, от качества его реализации зависит надежность и производительность системы.

Для того чтобы уменьшить время ожидания запроса, при арбитраже PCI используется такой подход: запрос должен выполняться за определенное время.

Это означает, что мастер шины должен осуществлять арбитраж для каждого запроса, который выполняется на шине. Шина PCI использует центральную схему арбитража, где каждый агент мастера имеет уникальный сигнал запроса (REQ#) и сигнал разрешения (GNT#). Для получения доступа к шине используется простой прием "рукопожатие", типа "запрос — ответ". Арбитраж на шине "скрытый", это означает, что он выполняется во время предыдущего запроса таким образом, чтобы для арбитража не использовались никакие циклы шины PCI, за исключением случая, когда шина находится в состоянии "свободно".

При определении алгоритма арбитража следует установить значение для гарантирования заданного времени ожидания в наихудшем случае. Тем не менее, т. к. алгоритм арбитража не является частью спецификации шины, то разработчики могут произвольно модифицировать его, но при этом они должны обеспечить соблюдение требований времени ожидания выбранных ими контроллеров ввода-вывода, а также для плат расширения. Шина позволяет агенту осуществлять транзакции back-to-back, а также предоставляет арбитру определенную гибкость в распределении запросов по приоритетам и весам. Арбитр может реализовывать схему любой сложности, при этом в любом такте должен быть активен только один сигнал GNT#.

Арбитр обязан осуществлять алгоритм "равноправия", чтобы избежать блокировок. В общем случае арбитр должен перейти к новому агенту, когда текущий мастер снимает его сигнал REQ#. "Равноправие" означает, что каждому потенциальному мастеру нужно предоставить доступ к шине, независимо от других запросов. Однако это не подразумевает, что все агенты обязаны иметь равный доступ к шине. Алгоритм равноправия не налагает специальных условий, когда сигнал LOCK# является активным (предполагая, что ресурс заблокирован). Если вместо блокировки ресурса система выполняет полную блокировку шины, то также может выполняться алгоритм равноправия. Тем не менее арбитр должен передать управление новому агенту, если первоначальная транзакция, которая предпринимала попытку монопольного доступа, завершена с типом "Retry" [9, 18].

Пример алгоритма арбитража системы

Рассмотрим пример возможной реализации алгоритма арбитража. Для реализации алгоритма равноправия схема арбитража основывается на двух уровнях. Каждому мастеру шины назначен один из уровней. В данном примере агенты первого уровня имеют больший приоритет по использованию ресурса шины по сравнению с агентами второго уровня (т. е. данные агенты обладают более малым временем ожидания и большей производительностью). Агенты второго уровня обладают равными правами доступа к шине относительно

друг друга. Однако агенты второго уровня в составе группы имеют равные права доступа к шине с одиночными агентами первого уровня. Система может назначить агентов на данный уровень следующим образом. Устройства мастеров шины типа видео, ATM или FDDI назначаются на первый уровень, устройства типа SCSI, ЛВС или стандартных мастеров шины расширения будут назначены на второй уровень. Рис. 5.3 иллюстрирует пример алгоритма арбитража равноправия, который использует два уровня арбитража.

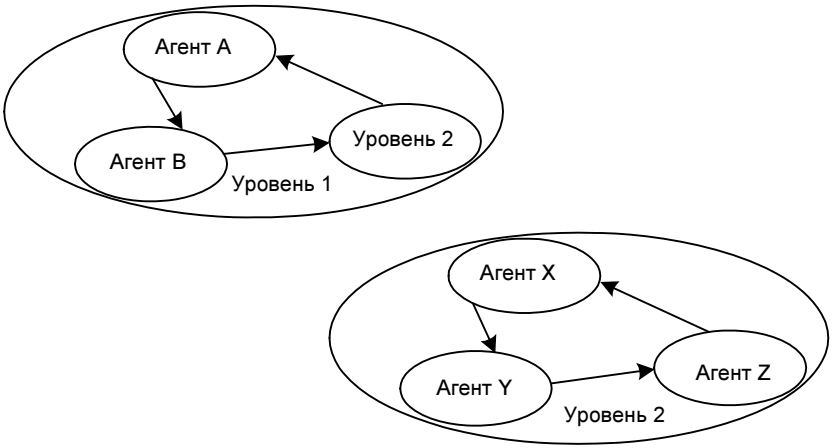


Рис. 5.3. Пример алгоритма арбитража

На первом уровне находятся агент A, агент B и уровень 2. Уровнем 2 обозначен следующий агент на уровне 2, запрашивающий доступ к шине. На втором уровне находятся агент X, агент Y и агент Z. Рассмотрим случай, когда выставлены сигналы REQ# всех агентов на уровнях 1 и 2, ни один из агентов не планирует снимать свой сигнал. Пусть агент A — следующий агент для доступа к шине уровня 1, а агент X — следующий для доступа к шине уровня 2. В этом случае предоставление агентам доступа к шине производилось бы в следующем порядке:

агент A, агент B, уровень 2 (агент X),

агент A, агент B, уровень 2 (агент Y),

агент A, агент B, уровень 2 (агент Z),

и т. д.

Пусть выставлены сигналы REQ# только двух агентов: агента B и агента Y. В этом случае порядок был бы следующим:

агент B, уровень 2 (агент Y),

агент B, уровень 2 (агент Y).

С целью оптимизации алгоритма арбитража разработчики могут балансировать потребности высокопроизводительных агентов типа видео, ATM или FDDI с низкопроизводительными устройствами шины. Разработчики могут построить алгоритм иначе, например, поместить на арбитражный уровень 1 только мультимедийные устройства, а на уровень 2 поместить FDDI (или ATM) и SCSI. В первом примере достигается высокий уровень производительности системы, во втором примере достигается наиболее низкое значение времени ожидания без возможных состояний "зависания". Производительность системы может быть сбалансирована выделением определенного количества полосы пропускания шины каждому агенту точным назначением каждого мастера на арбитражный уровень и соответствующим программированием таймера времени ожидания каждого агента [9].

Протокол сигналов арбитража

Запрос на использование шины агентом осуществляется путем установки сигнала REQ#. Агенты должны применять сигнал REQ# только для указания необходимости использования шины. Запрещается использовать сигнал REQ# для парковки шины. Под *парковкой шины* понимается закрепление агента на шине и предоставление ему полного контроля. Если парковка шины реализована, то таким агентом является арбитр, который назначается владельцем по умолчанию. Когда арбитр решает, что агент может использовать шину, то он устанавливает в активное состояние сигнал GNT# агента.

Перевод в неактивное состояние сигнала агента GNT# арбитром допускается на любом такте. Перед началом транзакции агент должен проверить состояние сигнала GNT# по переднему фронту синхроимпульса. Мастеру разрешено начинать транзакцию, если выставлен сигнал GNT# и шина находится в состоянии "свободно" независимо от состояния сигнала мастера REQ#. Запрещается продолжение транзакции при неактивном состоянии сигнала GNT#. Переход сигнала GNT# из активного состояния в неактивное должен выполняться в соответствии со следующими правилами:

1. Совокупность неактивного состояния сигнала GNT# и активного состояния сигнала FRAME# допускает продолжение транзакции.
2. Перевод в неактивное состояние одного сигнала GNT# может совпадать по времени с установкой другого сигнала GNT#, при условии, что шина не находится в состоянии "свободно". В противном случае потребуется один такт задержки между переводом в неактивное состояние сигнала GNT# и установкой другого сигнала GNT#, иначе возникнет одновременное использование адресных линий, и в итоге мастер должен будет выставить сигнал IDSEL.

3. Условие неактивности сигнала FRAME# определяет следующие правила. Сигнал GNT# может быть выставлен в любой момент времени в порядке обслуживания высокоприоритетным[1] мастером, или при отклике на соответствующий сигнал REQ#, переходящий в неактивное состояние.

На рис. 5.4 представлена диаграмма сигналов основного арбитража. Для наглядности используются два агента, чтобы показать, как арбитр может чередовать запросы шины.

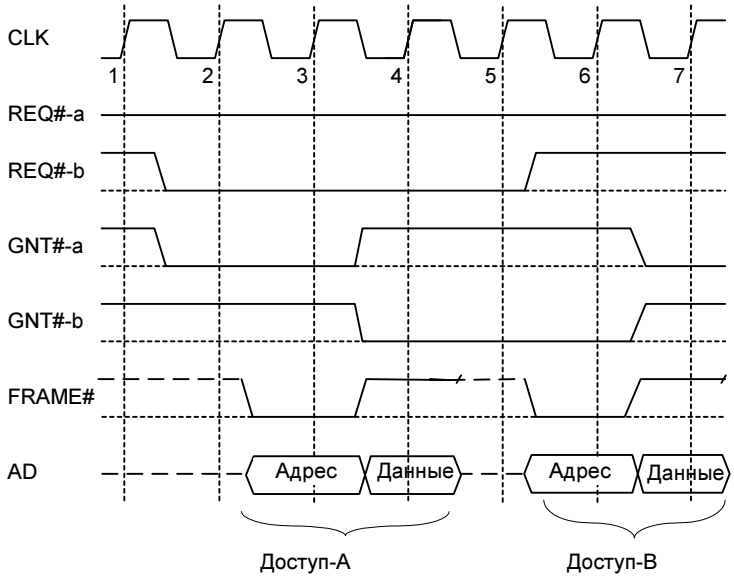


Рис. 5.4. Основной арбитраж шины

Запрос на использование шины осуществляется установкой сигнала REQ# до или во время такта 1. Агент A получает разрешение на доступ к шине, т. к. сигнал GNT#-a переводится в активное состояние на такте 2. Агент A может начать транзакцию в такте 2, т. к. сигналы FRAME# и IRDY# были установлены в неактивное состояние, а сигнал GNT# переведен в активное. Транзакция агента A начинается, когда на такте 3 выставляется сигнал FRAME#. Поскольку агент A решает выполнить другую транзакцию, то он оставляет сигнал запроса REQ#-a в активном состоянии. На такте 3 арбитр определяет, что далее следует агент B, и поэтому устанавливает в такте 4 сигнал GNT#-b в активное состояние, а сигнал GNT#-a — сбрасывает. На такте 4 агент A выполняет транзакцию и шина освобождается. Все агенты шины PCI могут оп-

[1] Высокий приоритет здесь не подразумевает фиксированный приоритетный арбитраж, это относится к агенту, который своевременно выиграл арбитраж в установленное время.

ределить окончание текущей транзакции по неактивным сигналам FRAME# и IRDY#. На такте 5 агент B захватывает шину (т. к. сняты сигналы FRAME# и IRDY#) и выполняет вою транзакцию на такте 7.

Неактивное состояние сигнала REQ#-b и активное сигнала FRAME# на такте 6 указывают, что агенту B требуется одиночная транзакция. Арбитр разрешает выполнять следующую транзакцию агенту A, т. к. его сигнал REQ# находится в активном состоянии.

Данный мастер шины не снимает сигнал REQ#, если ему требуются дополнительные транзакции. Арбитр предоставляет шину данному мастеру при отсутствии других активных запросов или если этот мастер имеет самый высокий приоритет.

Относительно парковки шины существуют некоторые рекомендации. Если сняты все сигналы REQ#, рекомендуется не снимать сигнал GNT# текущего мастера, чтобы закреплять (парковать) шину за различными мастерами, пока шина переходит в состояние "свободно". Если снят сигнал GNT# текущего мастера шины, то продолжительность транзакции ограничена значением таймера времени ожидания. При ограничениях данного таймера мастер должен повторно проходить алгоритм арбитража на шине, чтобы избежать пустых временных затрат. Рекомендуется не снимать сигнал GNT# текущего мастера (если сняты все сигналы REQ#), пока шина не перейдет в состояние "свободно". Арбитр может закреплять шину за любым агентом при условии, что шина находится в состоянии "свободно" и все сигналы REQ# находятся в неактивном состоянии [9].

Сигнал GNT# необходим агенту для запроса одиночной транзакции. Если агент желает сделать другой запрос, он должен сохранять активным сигнал REQ#. Агент может установить сигнал REQ# в неактивное состояние в любой момент времени, но арбитр может интерпретировать это как ситуацию, в которой агенту больше не требуется использования шины, и установить в неактивное состояние сигнал GNT#. Агент должен установить сигнал REQ# в неактивное состояние на том же такте, на котором активен сигнал FRAME#, при условии, что агенту требуется выполнить одиночную транзакцию. Когда транзакция завершена целевым устройством (активен сигнал STOP#), то мастер не должен снимать сигнал REQ# как минимум два такта. Первый из них требуется для перевода шины в состояние "свободно" (в конце транзакции, когда становится активным сигнал STOP#). Второй такт должен быть на такт раньше или позже после перехода в состояние "свободно". Это позволяет другому агенту использовать интерфейс, пока предыдущее целевое устройство готовится к следующему запросу.

Арбитр может принять решение, что текущий мастер поврежден, если тот не начал запрос по истечении 16 тактов после того, как был установлен в актив-

ное состояние сигнал GNT# (сигнал REQ# также выставлен). Арбитру разрешается игнорировать сигнал REQ# любого поврежденного мастера, допускается сообщать об этом состоянии системе. Арбитр может снять сигнал GNT# для обслуживания более приоритетного агента. Мастер, запросивший использование шины и не выставивший сигнал FRAME#, пока шина находилась в состоянии "свободно", может потерять очередь на шине. В сильно загруженной системе такой мастер может "зависнуть", потому что арбитр может предоставить шину другому агенту. Чтобы гарантированно получить доступ к шине, мастер должен выставить сигнал FRAME# на первом возможном такте, при условии, что сняты сигналы FRAME# и IRDY# и выставлен сигнал мастера GNT#[9].

Быстрые транзакции back-to-back

Существуют два типа быстрых транзакций back-to-back, которые могут быть инициированы мастером: транзакции, предназначенные тому же агенту, и транзакции, предназначенные другому агенту. Быстрые транзакции back-to-back разрешены на шине PCI в тех случаях, когда надо предотвратить одновременное управление линиями TRDY#, DEVSEL#, STOP# или PERR#.

Первый тип быстрой транзакции back-to-back предполагает ответственность мастера за отсутствие одновременного управления линиями. Мастер может удалять состояние "свободно" между транзакциями, при условии отсутствия одновременного управления. Данное условие может быть выполнено, когда для одной цели предназначаются две транзакции записи. Для этого типа быстрой транзакции back-to-back необходимо, чтобы мастер "знал" границы адресов потенциальной цели, иначе может возникнуть одновременное управление линиями. Данный тип быстрых транзакций back-to-back необязателен для реализации мастером, но должен дешифрироваться целевым устройством. Цель должна быть в состоянии обнаружить установку сигнала FRAME# от одного мастера без перехода шины в состояние "свободно".

Второй тип быстрой транзакции back-to-back предполагает ответственность цели за отсутствие одновременного управления линиями. Бит поддержки быстрой транзакции back-to-back в регистре статуса может быть установлен аппаратно только в том случае, если устройство отвечает следующим двум требованиям:

- целевое устройство не должно пропустить начало транзакции шины или потерять адрес, когда одна транзакция следует за другой без состояния "свободно" на шине. Другими словами, целевое устройство должно быть в состоянии отрабатывать состояние шины, начиная от заключительной передачи данных (снят сигнал FRAME# и выставлен сигнал IRDY#) и до фазы адреса (выставлен сигнал FRAME# и снят IRDY#) для последователь-

ных тактов. Цель должна отслеживать состояние шины независимо от того, выбрана она или нет на какой-либо из этих транзакций;

- цель должна избегать конфликтов для сигналов DEVSEL#, TRDY# STOP# и PERR#. Если цель не в состоянии установить сигнал DEVSEL# за минимальное по возможности время, то данное требование считается выполненным. Если цель реализует дешифрацию с нулевым временем задержки, то она должна задержать установку этих четырех сигналов в течение одного такта, за исключением следующих условий:
  - транзакции шины предшествовало состояние "свободно". Следовательно, данная транзакция не является транзакцией back-to-back;
  - цель управляла сигналом DEVSEL# на предыдущей транзакции шины, т. е. данная транзакция является транзакцией back-to-back, предназначенной для той же цели, что и предыдущая транзакция.

Задержка установки сигнала DEVSEL# для предотвращения одновременного управления линиями на быстрых транзакциях back-to-back не влияет на скорость дешифрации, указанную в регистре статуса.

Выполнение быстрых транзакций back-to-back требует от мастера реализации бита **Fast Back-to-Back Enable** в регистре команд. Этот бит имеет значение только для мастеров шины. Бит доступен для чтения и записи и его реализация не обязательна. Когда он установлен в единицу, мастер шины может начинать транзакцию PCI, используя синхронизацию back-to-back без указания адреса целевого устройства, взяв его из предыдущей транзакции записи. Если этот бит установлен в ноль или вообще не реализован, мастер может выполнять быстрые транзакции back-to-back только при условии, что новая транзакция будет выполняться для той же цели, что и предыдущая. Данный бит может быть установлен конфигурационным ПО после проверки состояния битов разрешения транзакций back-to-back (бит **Fast Back-to-Back Capable**) всех целей на шине.

Механизм быстрых транзакций back-to-back первого типа не позволяет выполнять такие транзакции для различных целей, в то время как механизм второго типа допускает быстрые транзакции для различных устройств.

Если целевое устройство неспособно обеспечить оба вышеуказанных условия, то оно не должно реализовывать данный бит вообще, тогда при чтении статусного регистра автоматически будет возвращаться ноль.

Быстрые транзакции back-to-back позволяют агентам использовать полосу пропускания шины более эффективно. Рекомендуется, чтобы цели и мастера реализовывали поддержку данной возможности, учитывая малые накладные расходы. При всех других условиях мастер должен вставить как минимум одно состояние "свободно" шины. Между транзакциями различных мастеров всегда существует по крайней мере одно состояние шины "свободно". Таким

образом, мастер обязан инициировать состояние "свободно", если не выполнены требования быстрой транзакции back-to-back или при смене мастера шины.

Если при выполнении транзакции back-to-back сигнал GNT# снят в последней фазе данных, то мастер теряет доступ на шину и должен предоставить ее другому мастеру. Последняя фаза данных выполняется при снятии сигнала FRAME# и активности сигналов IRDY# и TRDY# (или STOP#). Новый мастер начинает следующую транзакцию на том же такте, на котором передавались последние данные в предыдущей транзакции.

Агенты, не участвующие в выполнении быстрых транзакций back-to-back, могут не определять промежуточные границы транзакции и использовать только сигналы FRAME# и IRDY#. Мастера и цели должны определять эти границы только во время быстрых транзакций back-to-back. После завершения последней транзакции все агенты обнаружат состояние "свободно". Устройства, которые поддерживают механизм второго типа, должны быть в состоянии определить завершение всех транзакций PCI, а также распознавать все фазы адреса.

На такте 3 рис. 5.5 мастер выполняет запись, на такте 4 выполняется фаза адреса следующей транзакции. Цель должна распознать сигнал FRAME# на такте 4, т. к. предыдущая транзакция завершена на такте 3; в противном случае она потеряет адрес следующей транзакции [9].

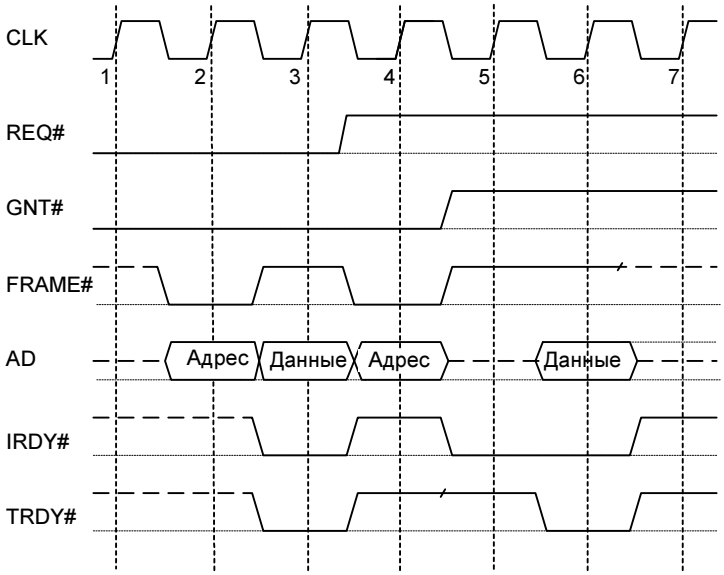


Рис. 5.5. Арбитраж транзакции back-to-back

Устройства должны быть в состоянии дешифрировать операции back-to-back, чтобы определить, являются ли они целью текущей транзакции. Поддержка этой функции мастером необязательна.

Парковка арбитража

Термин *парковка* подразумевает разрешение арбитру перевести в активное состояние сигнал GNT# для выбранного агента, если в данный момент времени ни один из агентов не использует и не запрашивает шину. Арбитр может выбирать владельца по умолчанию любым способом (закрепленного, последнего используемого и т. д.), либо вообще не выбирать ни одного, назначая себя в качестве владельца по умолчанию. Если арбитр переводит в активное состояние сигнал агента GNT# и шина находится в состоянии "свободно", то этот агент должен включить свои линии AD[3l::00], C/BE[3::0]# и на один такт позже выходные буферы PAR, не больше чем за 8 тактов, хотя рекомендуемое значение составляет 2—3 такта. Устройствам, которые поддерживают 64-битное расширение, разрешается парковка линий AD[63::32], C/BE[7::4]# и PAR64. Агент не обязан включать все буферы за один такт. Данное требование гарантирует нормальное функционирование шины при закреплении ее арбитром за некоторым агентом. Если шина не закреплена ни за одним из агентов, то шиной будет управлять устройство центрального ресурса, в которое встроен арбитр.

Агент теряет доступ к шине при снятии арбитром сигнала агента GNT#, если шина находится в состоянии "свободно", за исключением одного случая. В этом случае арбитр снимает сигнал GNT# агента, выставившего сигнал FRAME#, и мастер продолжает транзакцию. В противном случае агенту придется перевести в третье состояние линии AD[3l::00], C/BE# [3::0] и на один такт позже PAR. В отличие от вышеупомянутого варианта, агент должен отключить все буферы за один такт, чтобы избежать одновременного управления линиями другим агентом.

Подводя итог вышеизложенному, минимальные задержки для арбитража на шине PCI от состояния "свободно" должны быть следующими:

- агент закреплен (запаркован): ноль тактов — для закрепленного агента, два такта — для остальных;
- агент не закреплен: один такт для любого агента.

Когда шина закреплена за агентом, ему разрешено начинать транзакцию без установки сигнала REQ# (мастер может начать транзакцию, если шина находится в состоянии "свободно" и выставлен сигнал GNT#). Если агенту требуется выполнить несколько транзакций, то он должен установить сигнал REQ#, чтобы сообщить арбитру о своих намерениях. Когда мастеру требуется только одна транзакция, то он не должен выставлять сигнал REQ#; в про-

тивном случае арбитр установит сигнал GNT#, когда не требуется использование шины [9].

Методы оптимизации на шине PCI

Сигнал разрешения обращения к байтам и маршрут байтов

Маршрут байта подразумевает комбинацию линий данных, по которым могут передаваться данные, и определяется спецификацией процессора.

Протокол шины не поддерживает автоматической установки разрядности шины на 32-разрядных и меньше запросах. Это связано с тем, что автоматическое изменение разрядности позволяет устройству прерывать транзакцию, если доступ осуществляется в младшие для цели области. Например, к 8-разрядному устройству производится 16-битный запрос. Устройство передает младшие 8 битов и требует, чтобы для выполнения запроса мастер передал старшие 8 битов данного 16-битного доступа по младшим линиям. Все PCI-устройства соединяются с младшими 32 линиями для адресной дешифрации, следовательно, в случае необходимости устройство должно обеспечить управление каждым байтом. Другим выходом управления данными и сопоставление их с байтом является драйвер. Автоматическая установка разрядности шины поддерживается протоколом, если мастер производит 64-битный запрос к 32-битной цели. В этом случае цель не указывает, что она может делать 64-битную передачу данных, и мастер должен завершить текущую транзакцию, используя младшие 32 линии.

В фазе данных сигнал C/BE# функционирует в качестве сигнала разрешения обращения к байтам. Назначение этого сигнала — указать, по каким линиям передаются данные. Таким образом, он обеспечивает обращение к любому байту внутри двойного слова. Соотношение простое: сигналу C/BE#[0] соответствуют линии AD[0::7], C/BE#[1] — AD[8::15], C/BE#[2] — AD[16::23], C/BE#[3] — AD[24::31].

Если цели необходимо указать конкретные байты, то она должна ждать установку сигнала разрешения обращения к байтам C/BE# на каждой фазе данных; в противном случае цель должна передать все 4 байта. Этот сигнал активен в течение всей фазы данных, независимо от состояния линии IRDY#. Если цель поддерживает чтение с упреждением (бит 3 установлен в регистре базового адреса памяти), то она должна также возвратить все данные[2], независимо от сигнала C/BE#.

---

[2] Для 32-битной передачи данных это означает 4 байта в фазе данных, для 64-битной передачи — 8 байтов в фазе данных.

Протокол PCI допускает любую непрерывную или состоящую из нескольких несмежных участков комбинацию сигнала разрешения обращения к байтам. Если неактивен не один из сигналов C/BE#, цель доступа должна завершить фазу данных, выставив сигнал TRDY# и обеспечив контроль по четности, если транзакция является запросом чтения. При этом состояние цели не должно изменяться. Для транзакции чтения это означает, что не изменятся данные и состояние. Генерация и проверка контроля по четности для 32- и 64-битных передач данных одинаковы, независимо от состояния сигнала C/BE#.

При некоторых условиях цель не сможет правильно интерпретировать последовательности, состоящие из нескольких несмежных участков (например, если целью являются мост шины расширения, служащий интерфейсом к 8- и 16-битным устройствам). В этой ситуации мосты шины расширения могут дополнительно сообщать о запрещенных последовательностях как об ошибках с помощью сигнала SERR# или разбивать транзакцию на меньшие транзакции, которые являются допустимыми для данного агента. Цель транзакции ввода-вывода должна выполнить завершение типа "Target-Abort", если она не в состоянии выполнить доступ, определенный сигналом разрешения обращения к байтам [9].

Управление шиной и оборотный цикл

Некоторые сигналы могут управляться различными агентами, например, сигнал FRAME#. Для передачи управления от одного агента другому предусмотрен *оборотный цикл*. Данный цикл необходим, чтобы избежать одновременного управления линией, когда один агент прекращает управление сигналом, а другой агент начинает. Цикл обозначен на диаграммах как две стрелки, указывающие одна в хвост другой. Оборотный цикл происходит в различное время для различных сигналов. Например, сигналы IRDY#, TRDY#, DEVSEL#, STOP# и ACK64# используют фазу адреса в качестве своего оборотного цикла. Сигналы FRAME#, REQ64#, C/BE[3::0]#, C/BE[7::4]#, AD[31::00] и AD[63::32] используют в качестве своего оборотного цикла состояние "свободно" между транзакциями. Оборотный цикл для сигнала LOCK# занимает один такт после того, как его освобождает текущий владелец. Сигнал PERR# имеет оборотный цикл на четвертом такте после последней фазы данных, которая занимает три такта после оборотного цикла для адресных линий. Все адресные линии (включая [63::32], в случае если мастер поддерживает 64-разряда для линий данных), должны быть приведены в устойчивое (стабильное) состояние в течение каждой фазы адреса и фазы данных. Четные линии передачи байта, не участвующие в текущей передаче данных, должны передавать на шину одни и те же данные (хотя и не имеющие значения). Смысл этого заключается в вычислении контрольной суммы

по четности, а также для предотвращения состояний блокировки. В энергочувствительных приложениях, в целях уменьшения энергопотребления при переключениях шины, рекомендуется, чтобы линии передачи байта, не используемые в текущей фазе шины, управлялись с теми же самыми данными, что и в предыдущей фазе шины. Для обычных систем агент, управляющий линиями адреса, может произвольно управлять неиспользуемыми линиями передачи байта. Контроль по четности должен осуществляться для всех линий байтов, независимо от сигнала разрешения обращения к байтам [9].

Комбинирование, объединение и свертка

При некоторых условиях мосты, которые получают (записывают) данные, могут оптимизировать передачу данных на PCI путем комбинирования нескольких транзакций в одну, объединения байтов или слов в двойное слово, а также сверткой отдельных записей в память в отдельную шинную транзакцию. Таким образом снижается число транзакций и, следовательно, уменьшается время использования шины. Далее описана каждая из этих операций и рассмотрено их использование для мостов (главного, PCI-to-PCI, или моста стандартной шины расширения).

Комбинирование. *Комбинированием* называется объединение последовательных транзакций записи в память (отдельной фазы данных или блока, независимо от состояния сигнала разрешения обращения к байтам) в единственную транзакцию на шине PCI (используется линейное упорядочение в блоке). Объединять данные не требуется, но рекомендуется, когда данные для записи сохраняются в буфере. Производить объединение разрешено только в том случае, когда не изменено подразумеваемое упорядочение. Подразумеваемое упорядочение означает, что цель "видит" данные в том же порядке, в каком их генерировал исходный мастер. Например, последовательность записи двойных слов 1, 2, и 4 может быть преобразована в блок. Запись двойных слов 4, 3, и 1 не может быть объединена в блок и на PCI должны появиться три отдельных транзакции в том же самом порядке, в каком они произошли первоначально. Блоки могут включать фазы данных, которые не имеют выставленного сигнала разрешения обращения к байтам. Например, последовательность двойных слов 1, 2, и 4 может быть объединена в блок, в котором фаза данных 1 содержит данные, и сигнал разрешения обращения к байтам для двойного слова 1. Вторая фаза данных блока также содержит данные, и сигнал разрешения обращения к байтам для двойного слова 2. Для фазы данных 3 сигнал разрешения обращения к байтам не выставлен и данные этой фазы не имеют значения. Блок завершается фазой данных 4, которая содержит значимые данные и соответствующий сигнал разрешения обращения к байтам для двойного слова 4. Если цель не в состоянии обработать несколь-

ко фаз данных в одной транзакции, она заканчивает блочную транзакцию с типом "Disconnect" одновременно или после каждой фазы данных. Цель получает эти данные в том же самом порядке, в котором их генерировал мастер, независимо от того, была ли транзакция первоначально сгенерирована как блок или как ряд отдельных доступов фаз данных, которые были объединены в блок.

Объединение. *Объединение* применяется относительно байтов или слов и означает объединение последовательности индивидуальных записей в память (байта или слова) в отдельное двойное слово. Очевидно, объединение в пределах одного и того же двойного слова для 32-разрядных фаз данных или четверного слова (восемь байтов) для 64-разрядных фаз данных не требуется. Производить объединение рекомендуется, когда происходит буферизация данных для записи. Объединение байтов разрешено при условии, что байты в пределах фазы данных находятся в адресном интервале, доступном для упреждающего чтения. По сравнению с комбинацией, объединение может быть сделано в любом порядке (в пределах одной фазы данных), если каждый байт записан только один раз. Например, в последовательности, где байты 3, 1, 0 и 2 записываются в один и тот же адрес размером в двойное слово, мост мог бы объединить их в отдельную запись в память фазы данных на PCI с выставленными байтами разрешения 0, 1, 2, и 3 четырех индивидуальных транзакций записи. Но если последовательность, записываемая по одному и тому же адресу размером в двойное слово, была байт 1, и снова байт 1 (с теми же или другими данными), байт 2, и байт 3, мост не может объединить первые две записи в отдельную фазу данных, потому что в одно и то же место должно быть записано дважды два байта. Последние три транзакции могли быть объединены в отдельную фазу данных со снятым байтом разрешения 0 и выставленными байтами разрешения 1, 2 и 3. Объединение не может производиться при обращении в пространство ввода-вывода или в пространство отображенных адресов ввода-вывода, т. е. при обращении в адресное пространство, не доступное для упреждающего чтения.

Объединение и комбинирование могут происходить независимо друг от друга. Байты в пределах двойного слова могут быть объединены, а также объединенные двойные слова могут быть объединены с другими двойными словами, если позволяют условия. Устройство может осуществлять любую совокупность данных операций: либо объединение байтов, либо комбинирование, либо объединение байта и комбинирование, или не выполнять ни одну из них.

Свертка. *Сверткой* называется объединение последовательностей записей в память в одно и то же местоположение (адрес размером в байт, слово или двойное слово) и в одну транзакцию. Мостам PCI (главному, PCI-to-PCI или

стандартного расширения) производить свертку не разрешено, за исключением нескольких случаев. Например, транзакция записи в память, с выставленным сигналом разрешения обращения к байту 3 по адресу размером в двойное слово X, следуемая за доступом для записи в память по тому же самому адресу (X) размером в байт, слово или двойное слово, где выставлен сигнал разрешения обращения к байту 3, не может быть объединена в одну транзакцию PCI. Эти два доступа должны появиться на шине PCI как две отдельные транзакции.

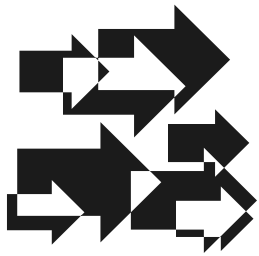
Комбинирование и объединение байтов для операций ввода-вывода и транзакций конфигурации не разрешены. Свертка данных транзакций любого типа (обращающихся в пространство конфигурации, ввода-вывода или пространство адресов памяти) не допускается, за исключением нескольких случаев. Считается, что если устройство не может произвести комбинирование записей в память, то оно было разработано неправильно. Если устройство не может допустить объединение байтов записи в память, то оно должно отметить себя как не доступное для упреждающего чтения. Устройство, которое идентифицировало себя как доступное для упреждающего чтения, должно обеспечить комбинирование (без изменения порядка следования) и объединение байтов (без свертки) записи. Устройству явно не требуется выполнять изменение порядка следования двойных слов или свертки данных. Интервал адресов, доступных для упреждающего чтения, может иметь побочные эффекты при записи, но может и не иметь побочных эффектов чтения. Мост (главной шины, PCI-to-PCI, или стандартной шины расширения) не может изменять порядок следования двойных слов в любом адресном пространстве. Мосты могут дополнительно произвести свертку данных в определенном адресном интервале, если драйвер устройства указывает, что отсутствуют побочные эффекты при свертывании [9].

Эффективность комбинирования, объединения и свертки

Мосты, которые сохраняют в буфере данные для записи в память, должны рассмотреть возможность выполнения комбинирования и объединения. Свертывание нескольких транзакций записи в память в единственную транзакцию на шине PCI не разрешено (кроме случаев, описанных выше). Комбинирование последовательностей 32-битных записей в память позволяет значительно повысить производительность, т. к. в результате создается единый блок. Например, процессор делает большое количество 32-битных записей в буфер изображения. При комбинировании главным мостом шины этих доступов в одну PCI-транзакцию шина PCI может не отставать от ведущей шины, которая работает быстрее и/или имеет большую разрядность, чем PCI.

Объединение байтов в пределах единственного двойного слова также повышает производительность, но не столь существенно, как комбинирование. Однако для невыровненных многобайтных передач данных объединение позволяет главному мосту объединять данные, расположенные с нарушением границ в отдельные двусловные транзакции записи в память. Это уменьшает число PCI-транзакций как минимум в два раза. Когда мост объединяет байты в двойные слова и затем комбинирует их в блок, число транзакций на шине PCI также снижается. С добавлением комбинирования последовательных двойных слов число транзакций на PCI может быть уменьшено еще больше. Объединение данных (в двойное слово) в пределах одной строки кэш-памяти практически не влияет на повышение производительности [9].

ГЛАВА 6



Служебные функции шины

Данная глава описывает дополнительные возможности шины, а также рекомендации по ее эффективному и оптимальному использованию. В разделе дополнительных операций рассмотрены служебные функции, применяемые в различных условиях. Выполнение некоторых критических операций требует обеспечения непрерывного процесса взаимодействия двух устройств. Шина PCI обеспечивает механизм монопольного доступа, который рассмотрен во втором разделе. Обеспечение помехоустойчивости и надежности без снижения производительности реализовано в виде проверки четности и механизма обнаружения и сообщения об ошибках четности и других ошибках системы. Раздел, описывающий функции обнаружения ошибок, содержит требования к устройствам и правила поведения системы при наличии каких-либо ошибок. В разделе, рассматривающем задержки на шине, приводится пример снижения производительности при неэффективной организации взаимодействия устройств на шине. Кроме того, рассмотрены рекомендации по расчету задержек и построению алгоритма арбитража. В заключительном разделе все правила взаимодействия устройств и выполнения дополнительных операций, требования максимальных значений задержек на шине PCI, управление служебными и другими линиями, описание правил перехода шины из состояния в состояние и основы протокола шины PCI сведены в итоговый перечень [9].

Дополнительные операции шины

Основной задачей шины является обеспечение взаимодействия устройств, т. е. основными являются операции по передаче данных. В процессе взаимодействия устройств необходимо предусмотреть ряд дополнительных операций, таких как, например, установление соединения. Такая операция носит название *выбор устройства* и предназначена для сообщения запрашивающе-

му устройству об установлении связи. Процессор может переходить в различные состояния, при которых невозможен обмен данными между устройствами. Для информирования всех устройств и агентов наиболее рационально использовать механизм широковещательных сообщений, на шине PCI этот механизм называется *специальный цикл*. В связи с аппаратными особенностями некоторые сигналы должны переводиться в активное состояние за период времени, превышающий допустимый предел. Способность агента к установке сигналов больше, чем за несколько тактов, называется *пошаговым режимом*. В любой стандартной последовательной архитектуре вычислительной системы особое место занимают прерывания; кроме того, шина PCI поддерживает подтверждение прерывания [9].

Выбор устройства

Взаимодействие различных устройств посредством шины PCI требует описания правил выбора устройства. Чтобы обратиться к конкретному устройству, необходимо его идентифицировать, т. е. отличить от других устройств. Данная операция называется "Выбор устройства". Рассмотрим последовательно все стадии данной операции.

Выбор какого-либо устройства на шине происходит по сигналу DEVSEL#. Этот сигнал управляется целью текущей транзакции, для сообщения того, что она отвечает на транзакцию. Данный процесс проиллюстрирован на рис. 6.1. Линия DEVSEL# может управляться во время одного, двух или трех тактов после окончания фазы адреса. Время нахождения сигнала DEVSEL# в активном состоянии для каждой цели зависит от значения, хранимого в ее регистре CONFIGURATION SPACE STATUS. Сигнал DEVSEL# должен быть выставлен одновременно или до фронта, на котором выставляются сигналы TRDY#, STOP# и данные во время транзакции чтения. Другими словами, цель должна выставить сигнал DEVSEL# (требуя транзакции) раньше или одновременно с сигнализацией любой другой отвечающей целью. После установки сигнал DEVSEL# не может быть снят, пока не завершится последняя фаза данных. Единственным исключением является аварийное завершение "Target-Abort".

Если сигнал DEVSEL# не выставлен ни одним агентом в течение трех тактов активного состояния сигнала FRAME#, сигнал DEVSEL# может выставить агент, производящий вычитающее дешифрирование. В случае отсутствия в системе такого агента, сигнал DEVSEL# не будет выставлен и мастер, не получив отклика, завершит транзакцию аварийным завершением "Master-Abort".

Цель должна производить полную дешифрацию перед управлением/установкой сигнала DEVSEL# или перед установкой любого другого сигнала отклика цели.

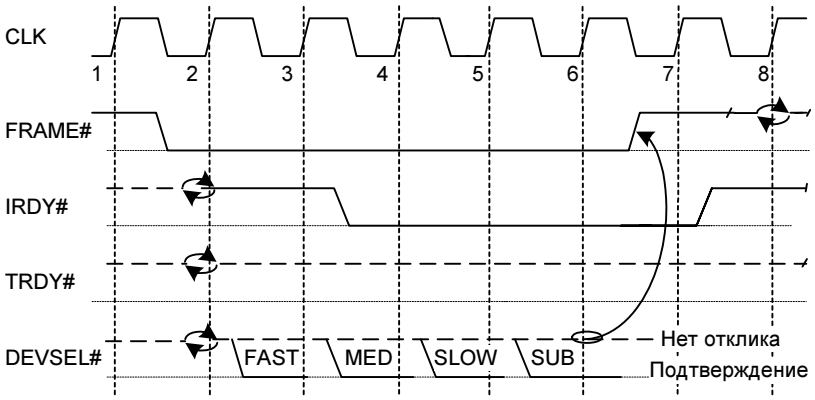


Рис. 6.1. Выбор устройства

Примечание

Запрещается управлять сигналом DEVSEL# до завершения дешифрации, т. к. это может привести к одновременному управлению линией двумя устройствами.

Цель должна принять сигналы по линиям AD и FRAME# прежде, чем может быть выставлен DEVSEL# на всех командах, кроме конфигурационных. Цель также должна принять сигналы с линий IDSEL, FRAME# и AD[1::0] прежде, чем сигнал DEVSEL# может быть выставлен в случае команды конфигурации.

Большинство целей в состоянии дешифрировать и выставлять DEVSEL# внутри одного или двух тактов активности FRAME# (FAST и MED на рисунке). Соответственно агент, выполняющий вычитающую дешифрацию, может обеспечить дополнительное устройство зависимым регистром конфигурации. Данный регистр определяет количество тактов (один или два), во время которых устройство производит выборку сигнала DEVSEL#. Использование этого регистра обеспечивает более быстрый доступ к шине. Функциональность ограничена самым медленным агентом на шине, осуществляющим положительную дешифрацию.

Если первый байт отображается в адресный интервал цели, то цель выставляет сигнал DEVSEL#, чтобы разрешить доступ. Но если мастер пытается продолжить доступ, выходя за границы ресурса, цель сообщает "Disconnect".

Если цели разрешается доступ к вводу-выводу и сигнал разрешения обращения к байтам указывает один или больше байтов для доступа вне адресного интервала цели, она должна завершить транзакцию с типом "Target-Abort". При наличии проблем с этим типом ввода-вывода устройство, выполняющее

вычитающую дешифрацию (мост шины расширения), может поступать следующим образом:

- выполнить положительную дешифрацию (путем включения байтовой карты) для адресов, на которых различные устройства совместно используют общие двойные слова, применяя сигнал разрешения обращения к байтам, чтобы обнаружить эту проблему и завершить транзакцию с типом "Target-Abort";
- передать весь доступ на шину расширения, где часть доступа, которая не может быть обслужена, будет просто удалена. (Это возможно если первая цель находится на шине расширения, а вторая — на шине PCI).

Специальный цикл

Процессор может переходить в различные состояния, при которых невозможен обмен данными между устройствами. Для информирования всех устройств, агентов, наиболее рационально использовать механизм широковещательных сообщений. Данный механизм реализован на шине PCI в виде специального цикла. Транзакция специального цикла не содержит адреса какого-либо устройства, а передается всем агентам. Агенты при получении данной команды не выставляют в ответ сигнал DEVSEL#, а определяют, применима ли она к ним. Во время фазы данных линии AD[31::00] содержат тип сообщения и дополнительное поле данных. Сообщение закодировано на младших 16 линиях, а именно AD[15::00]. Необязательное поле данных закодировано на старших 16 линиях AD[31::16] и используется не во всех сообщениях. Транзакция заканчивается с завершением "Master-Abort" [9].

Этот механизм может также использоваться для логической передачи сигналов между PCI-агентами, когда не требуется точной синхронизации или синхронизации физических сигналов. Поддержка команды *Special Cycle* необязательна.

Команда *Special Cycle* передается только в пределах одного сегмента шины и не выходит за пределы мостов PCI-to-PCI. Если мастеру необходимо генерировать команду *Special Cycle* на какой-либо входной шине в иерархии, то для этого он должен использовать команды конфигурационного доступа (запись) типа 1. Команды конфигурационного доступа (запись) типа 1 могут передаваться через мосты PCI-to-PCI в обоих направлениях с целью исполнения команды *Special Cycle* на любой шине в иерархии и имеют фиксированный размер — одна фаза данных. Соответственно, мастер должен указать шину, на которой ему необходимо сгенерировать команду *Special Cycle*. Он не может производить широковещательный запрос к одной шине в предположении, что запрос распространится на все шины.

Команда *Special Cycle* может дополнительно содержать данные, зависящие от передаваемого сообщения. Эти данные не интерпретируются PCI-секвенсором, а передаются им для обработки в аппаратные средства, присоединенные к секвенсору. В большинстве случаев явно адресованные сообщения должны быть обработаны в одном из трех физических адресных пространств без помощи команды специального цикла.

Использование данных, зависящих от сообщения, нарушает идею механизма и создает предпосылки для пропуска сигнала агентами. Но поскольку цели в данном случае функционируют в режиме приема, т. е. принимают и распознают сообщения, то на них лежит ответственность за обработку сообщения в минимальное время доставки (шесть шинных тактов). В противном случае они должны обеспечить любую необходимую буферизацию для сообщений, которые они принимают. Обычно эта буферизация ограничена одним мультивибратором. Это позволяет гарантировать достижение сигналом агентов. Тем не менее в некоторых случаях невозможно буферизовать или обработать все сообщения, которые могут быть получены.

Транзакция "Special Cycle", как и любая транзакция шины, содержит фазу адреса и фазу данных. Аналогично с другими командами, фаза адреса начинается с установки сигнала FRAME# и завершается при снятии сигналов FRAME# и IRDY#. Уникальность этой команды по сравнению с другими состоит в том, что при получении этой команды агент не выставляет сигнал DEVSEL#, и транзакция заканчивается с завершением "Master-Abort". "Master-Abort" является нормальным завершением для транзакций специального цикла, при этом именно для этого случая завершения "Master-Abort" сообщения об ошибках не передаются. Данная команда передается на все исполнительные устройства, и агенты, способные к обработке, принимают команду и обрабатывают запрос.

Во время фазы адреса адресные линии не содержат значимых данных, в отличие от линии команд. Адресные линии AD[31::00] не содержат никакого явного адреса, но должны находиться в одном из устойчивых уровней для корректной проверки честности. Во время фазы данных линии AD[31::00] содержат тип сообщения и дополнительное поле данных. Сообщение передается по младшим 16 линиям, а именно по линиям AD[15::00]. Необязательное поле данных передается по старшим 16 линиям AD[31::16] и используется не во всех сообщениях. Мастер, сгенерировавший команду *Special Cycle*, может вставлять состояния ожидания, как, впрочем, и при генерации любой другой команды, цель вставлять состояния ожидания не может. Сообщение и связанные с ним данные имеют значение только после установки сигнала IRDY#. Информация, содержащаяся внутри последующих фазах данных, и величина задержки зависят от сообщения. Если мастер вставляет состояние ожидания или выполняет несколько фаз данных, то он должен продлить

транзакцию, чтобы дать потенциальным целям достаточное для обработки сообщения время. Это означает, что мастер должен гарантировать, что доступ не будет завершен как минимум в течение четырех тактов (или больше) после того, как будут переданы последние значимые данные. Например, мастер удерживает сигнал IRDY# в неактивном состоянии в течение двух тактов для одной фазы данных команды *Special Cycle*. Поскольку мастер вставлял состояния ожидания, транзакция не может быть завершена с типом "Master-Abort" на пятом такте после сигнала FRAME# (один такт после вычитающей дешифрации) как обычно, а должна быть увеличена не менее чем на два дополнительных такта. Если транзакция содержит несколько фаз данных, мастер не может закончить команду *Special Cycle* как минимум в течение четырех тактов после последней фазы значимых данных. Тип сообщения или дополнительное поле данных укажут для потенциальных целей количество данных, которые будут переданы. Цель должна зафиксировать данные на первом такте сигнала IRDY#, который выставляется для каждой части передаваемых данных.

Во время фазы адреса линии C/BE [3::0]# содержат значение "0001" (код команды *Special Cycle*), линии AD[31::00] содержат случайные значения и должны игнорироваться. В течение фазы данных выставлены сигналы C/BE [3::0]#, а линии AD[31::00] имеют следующие значения: линии AD[15::00] содержат код сообщения, а линии AD[31::16] содержат данные, зависящие от сообщения.

Секвенсор шины начинает эту команду подобно всем другим и завершает ее с типом "Master-Abort". Аппаратная часть предоставляет всю информацию как для любой другой команды и запускает секвенсор шины. Когда секвенсор сообщает, что доступ завершен с типом "Master-Abort", аппаратная часть считает доступ завершенным. В этом случае бит **Received Master Abort** в конфигурационном регистре состояния не должен быть установлен. В самом коротком исполнении команда *Special Cycle* может выполниться за 5 тактов. Перед следующим доступом требуется один дополнительный такт для оборотного цикла. Следовательно, от начала Special Cycle до другого доступа требуется всего 6 тактов шины PCI [9].

Сообщения специального цикла

Всего может существовать 65536 сообщений. Общий объем сообщений составляет 64 Кбайта. Коды сообщений определены и представлены в табл. 6.1, зарезервированные коды предназначены для выделения PCISIG и использоваться не могут.

SHUTDOWN — широковещательное сообщение, показывающее, что процессор выключен.

Таблица 6.1. *Коды сообщений специального цикла*

Значение линий AD[15::0]	Тип сообщения
0000h	SHUTDOWN
0001h	HALT
0002h	Определяется архитектурой *x*86
0003h—FFFFh	Зарезервированные значения

HALT — широковещательное сообщение от процессора, показывающее, что он выполняет инструкцию остановки `Halt`.

Код сообщения 0002h является универсальным для всех процессоров и чипсетов семейства *x*86.

Значение линий AD[31::16] определяет специфическое значение сообщений специального цикла, устанавливаемое корпорацией Intel.

Использование или генерирование архитектурно-зависимых кодов не ограничивается только выдачей их компаниям, запросившим использование таких значений. Специфические коды могут использоваться любым производителем в любой системе. Эти коды обеспечивают существование системных специфических коммуникационных связей между скооперированными PCI-устройствами для решения задач взаимодействия, которые не могут быть решены стандартными типами циклов передачи данных [9].

Пошаговая передача адреса или данных

Способность агента к переводу сигнала в активное состояние за период, превышающий один такт, называется *пошаговым режимом* или *пошаговой передачей*. Все агенты должны поддерживать обработку пошаговой передачи IDSEL, поддержка генерирования не обязательна. Пошаговая передача разрешена только на выводах IDSEL для того, чтобы управлять этим сигналом по линии AD через нагрузочный резистор. IDSEL выбирается через комбинацию сигнала FRAME# и кода команды конфигурации типа 0. Рис. 6.2 иллюстрирует задержку устанавливаемого мастером сигнала FRAME# до управления всеми адресными линиями AD и связанными с ними сигналом IDSEL#. Мастеру разрешается (и требуется) управлять линиями AD и C/BE#, как только они будут ему предоставлены в собственность и шина находится в состоянии "свободно". При задержке установки сигнала FRAME# мастер может потерять право доступа на шину. Как и в случае с любым мастером, сигнал GNT# должен быть выставлен на переднем фронте прежде, чем будет выставлен сигнал FRAME#. Если сигнал GNT# был снят на участках, отме-

ченных на рисунке как "A", мастер должен немедленно перевести сигналы в третье состояние, потому что арбитр предоставил шину другому агенту (новый мастер имеет более высокий приоритет). Если сигнал GNT# был снят на участке "B" или "C", то сигнал FRAME# к этому времени уже будет установлен, и транзакция продолжится [9].

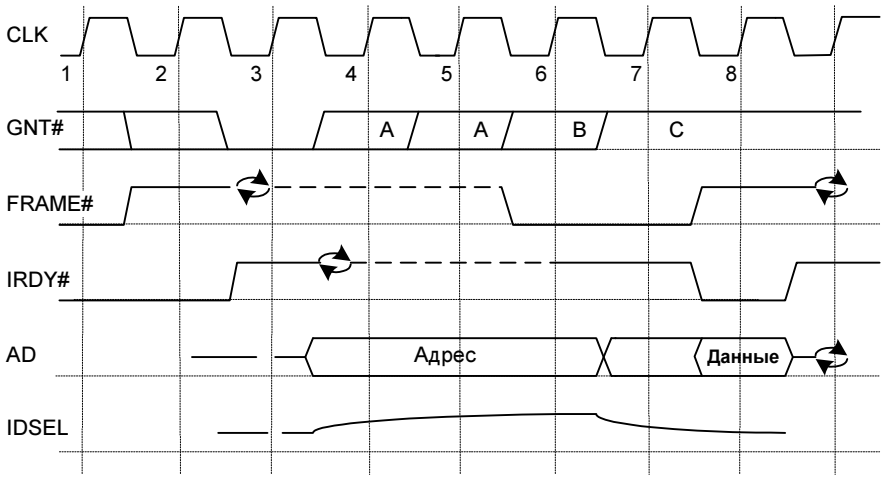


Рис. 6.2. Пошаговая передача адреса

Подтверждение прерывания

Прерывания в системе занимают особое место в любой стандартной последовательной архитектуре вычислительной системы. Шина PCI поддерживает цикл подтверждения прерывания, как показано на рис. 6.3. Этот рисунок иллюстрирует цикл прерывания процессора семейства *x*86 на шине PCI, где выставлен единственный сигнал разрешения обращения к байтам C/BE#. Этот сигнал определяет, какие байты включаются в транзакцию. Во время фазы адреса линии AD[31::00] не содержат значимый адрес, но должны управляться устойчивыми значениями, сигнал PAR разрешен и может быть выполнена проверка четности. Транзакция подтверждения прерывания не имеет никакого механизма адресации и неявно направлена к контроллеру прерывания в системе. Как определено в спецификации "PCI-to-PCI Bridge Architecture", команда *Interrupt Acknowledge* не посылается в другой сегмент шины PCI. Цикл подтверждения прерывания, так же как и любая другая транзакция, в которой сигнал DEVSEL# должен быть выставлен на один, два или три такта позже установки сигнала FRAME# для выполнения положительной дешифрации. Также может быть выполнена вычитающая дешифрация стандартным мостом шины расширения и вставлены состояния ожидания. Запрос

может быть завершен целью одним из способов завершения транзакции. Вектор прерывания должен быть возвращен во время установки сигнала TRDY#.

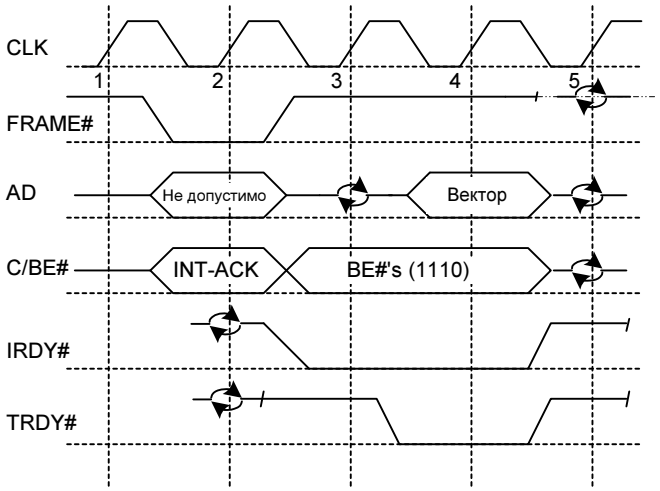


Рис. 6.3. Цикл подтверждения прерывания

В отличие от традиционного двойного цикла подтверждения прерывания, выполняемого контроллером 8259, шина PCI выполняет одиночный цикл. Преобразование из формата двухпроцессорных циклов в формат цикла PCI выполняется мостом путем отмены первого запроса подтверждения прерывания от процессора.

Монопольный доступ

Для обеспечения непрерывного обмена данными двух устройств был разработан механизм *монопольного доступа*. Реализация данного механизма требует отдельного сигнала, которым является сигнал LOCK#. Использование этого сигнала должно быть ограничено и разрешено только главным мостам, мостам PCI-to-PCI или мостам шин расширения. Все другие устройства не могут управлять сигналом LOCK#.

Главный мост может инициировать монопольный доступ для предотвращения состояния "зависания" шины или для обеспечения обратной совместимости с мостом шины расширения. Обеспечение совместимости подразумевает функционирование главного моста в качестве цели для монопольного доступа, инициированного мостом шины расширения. Всем другим агентам запрещено выполнение монопольного доступа к главному мосту.

Мост PCI-to-PCI может передавать монопольный доступ с его основной шины на вторичную, и ему запрещено выступать в качестве инициатора доступа. Мост PCI-to-PCI должен игнорировать сигнал LOCK# при функционировании в качестве цели на его вторичном интерфейсе.

Мост шины расширения может инициировать монопольный доступ только для обеспечения обратной совместимости. Это означает, что шина расширения аппаратно поддерживает механизм монопольного доступа. Мост шины расширения может поддерживать монопольный доступ в качестве цели, если доступ поддерживается шиной расширения; во всех других случаях сигнал LOCK# не имеет значения для моста [9].

Введение

Для выполнения некоторых критических операций необходимо обеспечить непрерывный процесс взаимодействия двух устройств. Шина PCI обеспечивает механизм монопольного доступа, который позволяет осуществлять обмен, несмотря на наличие блокированного доступа. Монопольный доступ состоит из нескольких этапов, каждый из которых выполняется в соответствии с определенными правилами. Такой тип доступа позволяет мастеру удерживать аппаратную блокировку при наличии других доступов без конфликтов с передачами данных. Примером монопольного доступа является передача видеоизображения в масштабе реального времени между двумя устройствами на некотором сегменте шины. Механизм основан на блокировании только того PCI-ресурса, для которого предназначался блокированный доступ. Такой тип монопольного доступа называется *блокировка ресурса*.

Блокированный доступ контролируется сигналом LOCK#. Сигнал LOCK# предназначен для использования мастерами шины и его наличие является признаком монопольного доступа. Управление LOCK# выполняется в соответствии со специальными правилами, связанными с сигналом GNT#. Агенты, не принимающие участие в монопольном доступе, могут продолжать выполнение обычных транзакций. Тем не менее для обеспечения совместимости арбитр может произвольно предоставить агенту, владеющему линией LOCK#, полный доступ на шину до освобождения линии. Такой доступ называется *полная блокировка шины*. При осуществлении блокировки ресурса цель доступа гарантирует монополию. Правила поведения мастера и цели для блокированных операций приведены в виде перечня.

Правила поведения мастера для поддержки сигнала LOCK#:

1. Мастер может получить доступ только к одному ресурсу во время блокированной операции.
2. Первая транзакция блокированной операции должна быть транзакцией чтения памяти.

3. Сигнал LOCK# должен быть выставлен на такте[1], следующем после фазы адреса, и должен оставаться в активном состоянии, чтобы удержать управление.
4. Линия LOCK# должна быть освобождена, если начальная транзакция блокированного запроса завершается с типом "Retry"[2].
5. Линия LOCK# всегда освобождается, если доступ завершен с типом "Target-Abort" или "Master-Abort".
6. Сигнал LOCK# должен быть снят между последовательными[3] блокированными операциями как минимум на один такт, пока шина находится в состоянии "свободно".

Правила поведения цели для поддержки сигнала LOCK#:

1. Мост, действующий как цель доступа, блокирует себя, когда сигнал LOCK# снят во время адресной фазы и выставлен на следующем такте.
2. После установки блокировки[4] мост находится в заблокированном состоянии до тех пор, пока не будут сняты сигналы FRAME# и LOCK# независимо от типа завершения транзакции.
3. Мосту не разрешено принимать какие-либо новые запросы (от любого интерфейса), пока он находится в заблокированном состоянии, кроме как от владельца сигнала LOCK#.

Старт монопольного доступа

Выполнению блокированных операций предшествует проверка состояния сигнала LOCK#, выполняющаяся агентом до установки сигнала REQ#. Если сигнал LOCK# выставлен, то он считается занятым для мастера (имеется в виду другой мастер, а не тот, которому требуется доступ). Сигнал LOCK# является незанятым при условии, что сняты оба сигнала FRAME# и LOCK#. Мастер не должен выставлять сигнал REQ#, пока занят сигнал LOCK#.

---

[1] Для *одиночного адресного цикла* или, иначе, *SAC* (Single Address Cycle) таким тактом служит такт после фазы адреса. Аналогично для *двойного адресного цикла* или, иначе, *DAC* (Dual Address Cycle) таким тактом является такт после первой фазы адреса.

[2] После установления блокировки мастер сохраняет управление сигналом LOCK# даже при завершении транзакции с типом "Retry" или "Disconnect".

[3] Последовательными называются back-to-back блокированные операции, а не продолжение текущей блокированной операции.

[4] Блокированная операция установлена, когда сигнал LOCK# снят во время фазы адреса, выставлен на следующем такте, и данные переданы во время текущей транзакции.

Во время ожидания установки сигнала GNT# мастер продолжает контролировать сигнал LOCK#. При некоторых условиях сигнал LOCK# может быть занят другим агентом, при этом мастер должен снять свой сигнал REQ#.

Управление сигналом LOCK# переходит к мастеру во время предоставления доступа к шине, при условии незанятости сигнала LOCK#. Мастеру разрешено выполнять монопольные операции, когда завершена текущая транзакция и на шине присутствует только сам мастер, управляющий сигналом LOCK#. Все остальные агенты не должны управлять сигналом LOCK# при его занятости, даже если они получают доступ к шине.

Начало монопольного доступа представлено на рис. 6.4. Во время фазы адреса (один такт для SAC или DAC) сигнал LOCK# снят. Таким способом запрашивается блокированная операция, которая указывается командой чтения памяти. После первой фазы адреса сигнал LOCK# должен быть выставлен на один такт после первой фазы адреса, что происходит на такте 3. Это необходимо, чтобы удержать цель в заблокированном состоянии. Такая последовательность действий позволяет текущему мастеру сохранять владение сигналом LOCK# после окончания текущей транзакции.

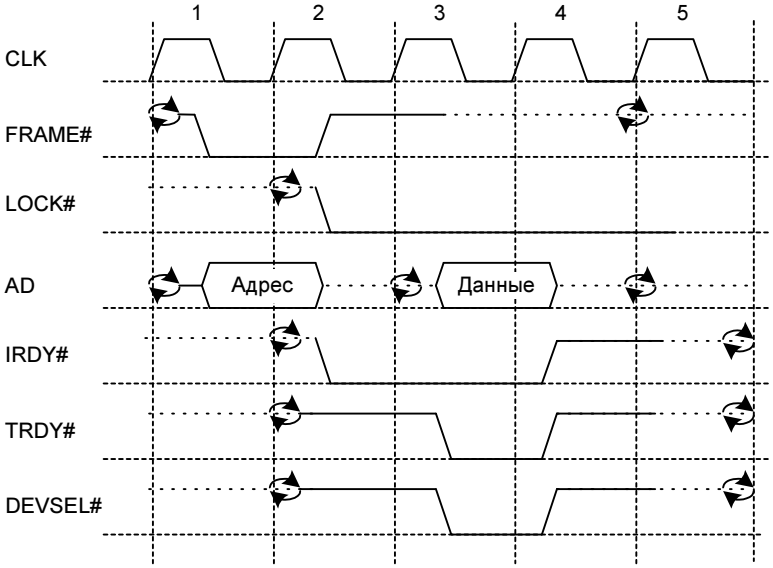


Рис. 6.4. Начало монопольного доступа

Монопольный доступ на шине не устанавливается до выполнения первой фазы данных (выставлены сигналы IRDY# и TRDY#). Если цель завершает первую транзакцию с типом "Retry", то мастер завершает транзакцию и освобождает линию LOCK#. Как только выполнена первая фаза данных с выстав-

ленными сигналами TRDY# и IRDY#, монопольный доступ установлен. После этого мастер удерживает сигнал LOCK# в активном состоянии до окончания блокированных операций или до появления ошибки ("Master-Abort" или "Target-Abort"). После того как установлен монопольный доступ, завершения транзакции с типом "Retry" и "Disconnect" являются допустимыми. Завершение целью транзакции с типом "Retry" или "Disconnect" после установления монопольного доступа указывается, что она (цель) в настоящее время занята и не в состоянии выполнить требуемую фазу данных. Цель примет доступ как только освободится, при этом она удерживает блокировку путем исключения всех других доступов. В такой ситуации мастер продолжает удерживать сигнал LOCK#. Установка сигнала LOCK# не препятствует выполнению обычных доступов к незаблокированным целям в том же сегменте шины. При блокировании моста транзакции для другого сегмента не могут проходить через мост.

Когда мост заблокирован, он может принимать запросы, только когда во время фазы адреса снят сигнал LOCK#. Это указывает на то, что транзакция является продолжением последовательности доступа мастером, который установил монопольный доступ. Если LOCK# выставлен во время фазы адреса, заблокированный мост завершит все доступы, выставляя сигнал STOP# при снятом TRDY# (завершение типа "Retry"). Заблокированная цель не освобождается до снятия сигналов FRAME# и LOCK# [9].

Выполнение задержанных транзакций

Блокированная транзакция может быть выполнена путем "завершения задержанной транзакции". При этом выполняются все правила управления сигналом LOCK#, за исключением того, что мост должен считать себя заблокированным при постановке запроса в очередь даже при отсутствии передачи данных. Такое состояние называется *блокировка цели*. В таком состоянии мост не ставит в очередь новые запросы на основном интерфейсе и завершает все запросы с типом "Retry". Мост блокирует свой вторичный интерфейс при установлении монопольной операции на вторичной шине. После этого мост начинает сканировать первичный интерфейс на предмет повторения исходного блокированного запроса. Блокировка цели переходит в состояние полной блокировки после повторения мастером блокированного запроса и передачи данных мостом. Начиная с этого момента, мост установил монопольный доступ.

Мост, функционирующий как цель, которая поддерживает монопольные доступы, должен принимать сигнал LOCK# на такте адреса и следующим за ним. Если мост выполняет среднюю или медленную дешифрацию, он должен зафиксировать сигнал LOCK# во время первой фазы адреса. В противном слу-

чае после выполнения дешифрации мост не сможет определить, является ли доступ блокированной операцией. Мост помечает себя как заблокированная цель, если сигнал LOCK# является снятым во время первой фазы адреса и выставлен на следующем такте. Если на такте после первой фазы адреса сигнал LOCK# снят, мост не указывает себя заблокированной целью и может отвечать на другие запросы [9].

Продолжение блокированных операций

Продолжение мастером блокированной операции представлено на рис. 6.5. Данная транзакция может быть как промежуточной, так и заключительной в монопольном доступе. Когда мастеру предоставляется доступ к шине, он начинает следующий монопольный доступ к цели, которую он предварительно заблокировал. Для продолжения блокированной операции сигнал LOCK# снимается во время фазы адреса. Заблокированное устройство принимает запрос и отвечает на него. На такте 3 (рис. 6.5) сигнал LOCK# выставляется для удержания цели в заблокированном состоянии, что позволяет текущему мастеру сохранить управление LOCK# после окончания текущей транзакции. Если мастеру необходимо продолжить монопольный доступ, то он сохраняет линию LOCK# в активном состоянии. По окончании блокированных операций мастер снимает сигнал LOCK# после завершения последней фазы данных, что происходит на такте 5 [9].

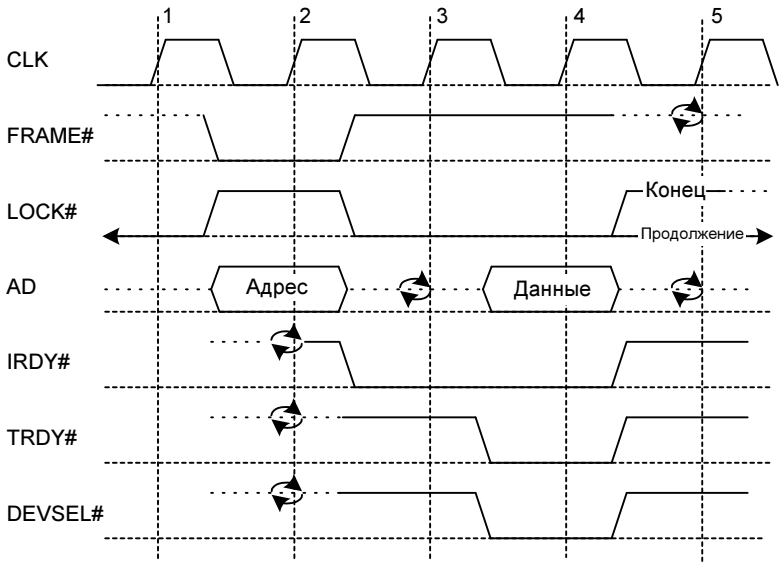


Рис. 6.5. Продолжение монопольного доступа

Доступ к заблокированному агенту

Попытка неблокированного доступа мастера к заблокированному агенту приведет к выставлению целью сигнала STOP#. В этом случае цель является заблокированной (полная блокировка или целевая блокировка), а сигнал LOCK# выставлен во время фазы адреса. Таким образом, цель завершит транзакцию с типом "Retry" без передачи каких-либо данных [9].

Завершение монопольного доступа

Во время заключительной транзакции блокированных операций сигнал LOCK# является снятым, так что цель примет запрос, и затем LOCK# будет снова выставлен до окончания блокированной транзакции. Мастер может снять сигнал LOCK# в любое время после окончания монопольного доступа. Рекомендуется, чтобы этот сигнал был снят одновременно с переводом в неактивное состояние сигнала IRDY# после завершения последней фазы данных блокированной транзакции. Прекращение управления линией LOCK# на любых других тактах может привести к лишнему завершению с типом "Retry". Заблокированный агент разблокирует себя по снятию сигналов LOCK# и FRAME#. Если мастеру необходимо выполнить два независимых монопольных доступа на шине, то между снятием и установкой сигналов FRAME# и LOCK# он должен пропустить как минимум один такт. Данное условие необходимо для того, чтобы любая заблокированная цель была освобождена после завершения монопольного доступа, т. к. по снятию сигналов FRAME# и LOCK# на одном такте агент должен разблокировать себя [9].

Полная блокировка шины

Блокировка PCI-ресурса может быть преобразована в полную блокировку шины. В такое состояние шина может перейти, если арбитр не предоставляет шину любому другому агенту во время нахождения сигнала LOCK# в активном состоянии. Если первый доступ блокированной последовательности завершен с типом "Retry", то мастер должен снять сигналы REQ# и LOCK#. Если первый доступ заканчивается нормально, то установлена полная блокировка шины и арбитр не предоставит шину какому-либо другому агенту. Арбитр мог предоставить шину другому агенту во время установления полной блокировки шины. В этом случае арбитр должен отменить предоставление шины другому агенту для соблюдения семантики полной блокировки шины. Полная блокировка шины может оказывать достаточно сильное влияние на производительность системы, особенно на видеоподсистему. Это связано с тем, что во время выполнения полной блокировки шины не могут происходить другие, неблокированные операции [9].

Функции обнаружения ошибок

Шина PCI обеспечивает обнаружение и передачу сообщений об ошибках четности и других ошибках системы. Отдельные системы могут включать как устройства, для которых обнаруженные ошибки не представляют интереса (особенно ошибки четности), так и агентов, которые обнаруживают, сообщают и исправляют ошибки. Механизм сообщения об ошибках PCI позволяет исключить возможное влияние агентов, которые исправляют ошибки четности, на агентов, которые этого не делают. Для обеспечения такой гибкости генерация четности требуется на всех транзакциях всеми агентами. Обнаружение и сообщение ошибок требуется, с некоторыми исключениями, для некоторых классов PCI-агентов, как это перечислено далее.

Рассмотрение ошибок вынесено в следующие два раздела, охватывающие генерацию четности, обнаружение и передачу сообщений об ошибке. Каждый раздел объясняет, что является необязательным, а что требуется для каждой функции [9].

Генерация четности

На шине PCI предусмотрен механизм, позволяющий контролировать выполнение транзакций и передаваемые данные. В радиотехнике такой механизм обозначается термином *помехоустойчивость*. Этим механизмом является контроль по четности. Для контроля правильности выполнения команд шины линии команд проверяются на четность. Для контроля правильности передаваемых данных в проверку четности разрешено включать 4 байта (32 линии данных). Агент, который управляет линиями AD[31::00] в любой фазе транзакции шины, также ответственен за управление линией PAR. Описанные далее требования применяются также при использовании 64-битной шины.

Во время фаз адреса и данных четность проверяется для линий AD[31::00] и C/BE[3::0]# независимо от того, несут или нет все линии значимую информацию. По линиям данных фактически не передаются данные, тем не менее на них должны присутствовать стабильные (хотя и бессмысленные) данные, т. к. они включены в вычисление четности. Во время транзакций конфигурации, специального цикла, или подтверждения прерывания некоторые (или все) линии адреса не определены, но все равно должны быть доведены до устойчивых значений и включены в вычисление четности.

Операция проверки четности функционирует согласно следующим правилам:

- алгоритм вычисления (проверки) четности одинаков на всех PCI-транзакциях независимо от типа или формы;

- сумма логических единиц на линиях AD[31::00], C/BE[3::0]# и PAR равняется четному числу;
- четность проверяется обязательно; это должно быть выполнено всеми PCI-совместимыми устройствами.

На любой фазе шины линия PAR управляется тем же агентом, который управляет линиями AD[31::00]. Сигнал на линии PAR отстает от сигнала соответствующей линии адреса или данных на один такт. Рис. 6.6 иллюстрирует транзакции чтения и записи с четностью. Мастер управляет линией PAR для фаз адреса на тактах 3 и 7. Цель управляет линией PAR в фазе данных на транзакции чтения (такт 5), в то время как мастер управляет сигналом PAR в фазе данных на транзакции записи (такт 8). Один такт отстает от другого, как и определено правилами. Линия PAR ведет себя подобно адресным линиям AD[31::00], включая состояния ожидания и оборотные циклы [9].

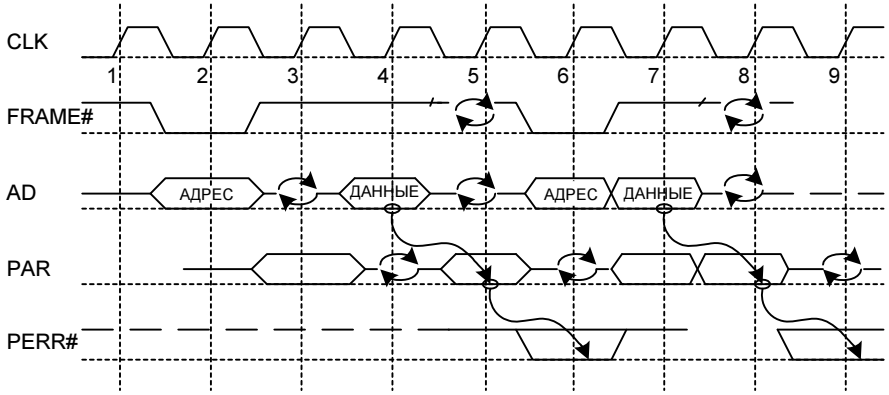


Рис. 6.6. Операция четности

Проверка четности

Для контроля за передаваемыми адресом и данными необходимо производить проверку четности. Все устройства должны проходить контроль по четности, за исключением устройств следующих двух классов, для которых данное требование является необязательным:

- устройств, предназначенных исключительно для использования на системной плате, например, для чипсета. Производители системы контролируют использование этих устройств, т. к. они никогда не будут применяться на платах расширения;
- устройств, не хранящих и не обрабатывающих никаких данных, которые характеризуют постоянное или остаточное состояние системы или состоя-

ние приложения; например, устройств взаимодействия с пользователем и устройств обработки видео- или звуковых сигналов. Эти устройства работают только с данными, которые являются временным представлением (например, с пикселами) постоянного или остаточного состояния системы или приложения. Следовательно, они не могут создать проблем, связанных с нарушением целостности системы в случае необнаруженного сбоя [9].

Ошибки четности адреса

Схемная логика устройства контроля по четности осуществляет проверку четности в одиночном адресном цикле и во всех фазах адреса двойного адресного цикла. В случае обнаружения ошибки устройство сообщает об этом посредством сигнала SERR# и специального бита в регистре состояния. Устройство должно установить бит **Detected Parity Error** (регистр состояния, бит 15) во всех случаях обнаружения ошибки четности адреса, сигнал же SERR# выставляется в других случаях, рассмотренных отдельно.

В случае обнаружения устройством ошибки четности адреса необходимо выполнение следующих условий: установлен бит **Parity Error Response** устройства (бит 6 регистра команд) и адресный дешифратор устройства указывает, что устройство выбрано. При выполнении этих условий устройство должно выполнить одно из следующих действий:

- разрешить выполнение транзакции и завершить ее, как будто ошибки адреса/команды не происходило;
- требовать транзакции и завершить ее с типом "Target-Abort";
- не требовать транзакции и позволить завершиться ей с типом "Master-Abort".

Ошибка в фазе адреса может произойти вследствие ошибки любого или всех адресных битов, битов команд или битов контроля четности. Устройства не могут определить, в каких битах фактически происходит ошибка. Следовательно, выполнение транзакции, которая содержала ошибку контроля по четности адреса, может привести к непредсказуемым последствиям. Цели не разрешено требовать транзакции и завершать ее с типом "Retry" только из-за ошибки контроля по четности адреса или ошибки контроля по четности данных для записи[5]. Ошибка контроля по четности не запрещает цели завершать транзакции с типом "Retry" по другим причинам [9].

[5] Цель проверяет четность данных только на транзакциях записи.

Сообщения об ошибках

Шина PCI обеспечивает обнаружение и передачу сообщений об ошибках. При этом ошибки разделяются на два типа: ошибка четности данных и все остальные системные ошибки. Сообщения об ошибках четности данных должны поступать через доступ и цепочку драйверов всякий раз, когда это возможно. Передача сообщения состоит из нескольких этапов: от цели к мастеру шины, затем драйверу устройства, после этого менеджеру устройства, далее операционной системе. Такая последовательность обеспечивает возможность восстановления при ошибках на любом уровне. Системные ошибки невозможно сопоставить со специфической цепочкой доступа, поэтому для сообщения о них используется отдельный системный сигнал ошибки.

PCI-устройства могут сообщить об ошибках контроля по четности данных посредством бита **Parity Error Response** (бит 6 регистра команд). Этот бит должен присутствовать во всех устройствах за исключением тех, которым не требуется проверять контроль по четности. Если бит **Parity Error Response** установлен, то устройства должны отвечать и отчитываться об ошибках контроля по четности данных для всех операций шины (за исключением тех, которые происходят во время транзакции специального цикла). Если бит **Parity Error Response** сброшен, то агент, который обнаруживает ошибку контроля по четности данных, должен игнорировать ошибку и выполнять транзакцию, как если бы контроль по четности был правилен. В этом случае не может происходить никакой специальной обработки ошибки контроля по четности данных. В системе сообщения об ошибке на шине PCI используются два сигнала (вывода) и два бита состояния, каждый из которых будет рассмотрен отдельно [9].

Сообщение по линии PERR#

Сигнал PERR# используется для сообщения об ошибках четности данных во всех транзакциях, кроме специального цикла. Ошибки четности данных, которые происходят во время специального цикла, передаются по линиям SERR#. Линия PERR# должна быть реализована всеми устройствами, кроме тех, которые не нуждаются в проверке четности.

Если включена поддержка сообщений об ошибке (установлен бит 6 регистра команд) и мастером обнаружена ошибка четности данных во время *транзакции чтения*, то мастер должен выставить сигнал PERR#. Если же включен отклик об ошибке (установлен бит 6 регистра команд) и целью обнаружена ошибка четности данных во время *транзакции записи*, то цель должна выставить сигнал PERR#. Мастера используют эту информацию, чтобы записать событие ошибки для драйвера устройства. Для мастера вывод PERR# работает как вход/выход, для цели только как выход.

При возникновении ошибки в фазе данных устройство должно выставить сигнал PERR# через два такта после выполнения фазы данных, как это показано на рис. 6.6. Если принимающий агент вставляет состояния ожидания, то агенту разрешено выставить сигнал PERR#, как только обнаружена ошибка четности данных. Кратко данный алгоритм выглядит следующим образом. Мастер выставляет данные, цель вставляет состояние ожидания. Если одновременно с состоянием ожидания (сигнал TRDY# снят) обнаруживается ошибка четности данных, то цель, не выставляя сигнал TRDY#, устанавливает сигнал PERR# через два такта после обнаружения ошибки. Такое же правило работает для мастера. Если мастер вставляет состояния ожидания во время транзакции чтения, то мастеру разрешено выставить сигнал PERR# на период двух тактов после того, как данные действительны (выставлен сигнал TRDY#), но до того, как данные переданы (выставлен сигнал IRDY#). Сигнал PERR# должен находиться в активном состоянии еще два такта после выполнения фазы данных (сигналы IRDY# и TRDY# выставлены). Необходимо учитывать, что мастер должен обеспечивать правильное формирование сигнала разрешения обращения к байтам во время каждого такта синхронизации каждой фазы данных для транзакций чтения и записи, независимо от состояния сигнала IRDY#. Если мастер выставляет сигнал PERR# до выполнения фазы данных чтения, он все равно должен выставить сигнал IRDY#, чтобы выполнить фазу данных. Если цель выставляет сигнал PERR# до выполнения фазы данных записи, она также должна установить сигнал TRDY#, чтобы выполнить фазу данных. Цель не может завершить фазу данных с завершениями транзакции типа "Retry", "Disconnect without Data", или "Target-Abort" после установки сигнала PERR#. Мастер получает информацию об ошибке четности данных в любое время установки сигнала PERR#. Но о том, что в фазе данных была ошибка, мастер узнает только через два такта после выполнения фазы данных. И мастерам и целям разрешено либо продолжить блочную транзакцию, либо остановить ее после обнаружения ошибки четности данных. Во время блочной транзакции, в которой несколько фаз данных выполнены без дополнительных состояний ожидания, сигнал PERR# будет допустим на соответствующих последовательных тактах и может быть выставлен на любых из них. Сигнал PERR# является подчиненным тристабильным сигналом, который "шинно" соединен со всеми PCI-агентами. Он должен активно переводиться к корректному значению на каждом заданном уровне такта агентом, получающим данные. В конце каждой операции шины сигнал PERR# должен находиться не в плавающем, а в высоком логическом состоянии. Такое управление должно обеспечиваться во время одного такта агентом, получающим данные, через два такта после оборотного цикла на линиях AD (например, такт 7 в рис. 6.6). Оборотный цикл линии PERR# происходит на один такт позже (такт 8 в рис. 6.6). Сигнал PERR# не может управляться для текущей транзакции, по крайней мере, три такта после фазы адреса (длительность которой составляет один такт для одиночного адресного цикла и два такта для двойного адресного цикла). Цель транзакции записи

не должна управлять никаким сигналом до окончательной установки сигнала DEVSEL#; например, для медленной дешифрации цель не должна управлять сигналом PERR# четыре такта после фазы адреса [9].

Сообщение по линии SERR#

Устройство может выставлять сигнал SERR# при условии разрешения (т. е. когда установлен бит 8 в регистре команд), и когда установлен бит устройства **Parity Error Response** (бит 6 регистра команд). Сигнал SERR# выставляется устройством при наличии одного из следующих событий:

- ❒ логика проверки четности устройства обнаружила ошибку в одиночном цикле адреса или любой фазе адреса двойного цикла адреса (независимо от цели);
- ❒ устройство отслеживает транзакции специального цикла, установлен бит **Special Cycles** (бит 3 регистра команд), и логика проверки четности устройства обнаруживает ошибку четности данных.

Сигнал SERR# может также использоваться, чтобы сообщить о других внутренних ошибках, которые могли бы привести к повреждению данных или сбою в системе. Необходимо учитывать, что передача сигналов на SERR# генерирует критическое системное прерывание (например, NMI или Machine Check) и поэтому фатальна. Следовательно, сигнал SERR# должен использоваться для сообщения о системных ошибках или ошибках четности крайне редко. Сигнал SERR# должен быть реализован всеми устройствами, кроме тех, которым не требуется проверять четность. Сигнал SERR# — это сигнал с открытым коллектором, который является "Проводным ИЛИ" со всеми другими PCI-агентами и, следовательно, может одновременно управляться несколькими агентами. Агент, сообщающий об ошибке по линии SERR#, управляет им один такт и затем переводит его в третье состояние. Логика схем с открытым коллектором не может гарантировать устойчивые сигналы на каждом переднем фронте синхроимпульса. Учитывая, что данный сигнал находится в активном состоянии один такт, то после установки сигнала SERR# его логическое значение должно быть принято неопределенным, пока сигнал не будет зафиксирован в снятом состоянии как минимум при двух поочередных передних фронтах синхроимпульса [9].

Бит состояния *Master Data Parity Error*

Бит **Master Data Parity Error** (бит 8 регистра состояния) устанавливается мастером, если установлен его бит **Parity Error Response** (бит 6 регистра команд) и происходит любое из следующих событий:

- ❒ мастер обнаруживает ошибку четности данных при транзакции чтения;
- ❒ мастер обнаруживает выставленный сигнал PERR# при транзакции записи.

Если бит **Parity Error Response** сброшен, то мастер не должен устанавливать бит **Master Data Parity Error**, даже при обнаружении ошибки четности или установке целью сигнала PERR#. Цель не может установить бит **Master Data Parity Error**.

Бит состояния *Detected Parity Error*

Бит **Detected Parity Error** (бит 15 регистра состояния) должен быть установлен устройством при обнаружении ошибки четности. Данный бит не зависит от состояния бита **Parity Error Response** (бит 6 регистр команд). Бит **Detected Parity Error** должен быть установлен устройством при наличии одного из следующих событий:

- ❒ логика проверки четности устройства обнаружила ошибку в одиночном цикле адреса или любой фазе адреса двойного адресного цикла;
- ❒ логика проверки четности устройства обнаружила ошибку четности данных, и при этом устройство является целью транзакции записи;
- ❒ логика проверки четности устройства обнаружила ошибку четности данных, и при этом устройство является мастером транзакции чтения.

Ошибки четности задержанных транзакций

Задержанные транзакции реализованы через особый механизм взаимодействия устройств, и, следовательно, ошибки четности данных таких транзакций должны быть обработаны в соответствии с определенными правилами. Основные требования к обработке ошибок четности данных, представленные в предыдущих разделах, применяются также и к задержанным транзакциям.

Ошибка четности данных может произойти во время любого из трех этапов задержанной транзакции: на шаге запроса мастера, на шаге выполнения цели, или на шаге выполнения мастера. Требования обработки ошибки изменяются в зависимости от этапа, на котором произошла ошибка. Ошибки, которые происходят во время фазы выполнения целью, являются специфическими для целевого устройства и обрабатываются устройством произвольно. Поведение устройства при ошибках, которые происходят во время шага запроса мастера или шага выполнения мастером, зависит от того, является ли задержанная транзакция транзакцией чтения[6] или транзакцией записи[7].

---

[6] Это могут быть команды *Memory Read*, *Memory Read Line*, *Memory Read Multiple*, *Configuration Read*, *I/O Read* или *Interrupt Acknowledge*.

[7] Это могут быть команды *Configuration Write* или *I/O Write*, но не команды *Memory Write and Invalidate* или *Memory Write*.

Во время транзакции чтения целевое устройство передает данные, и четность не проверяется, пока не выставлен сигнал TRDY#. Поэтому ошибка четности данных не может произойти во время фазы запроса мастера или любого последующего повторного запроса мастера при завершении с типом "Retry". Во время шага выполнения мастером транзакции чтения цель предоставляет данные и проверяет четность. Мастер проверяет четность и выставляет сигнал PERR# как для любой другой (не задержанной) транзакции.

Во время транзакции записи мастер предоставляет данные для записи и должен выставить сигнал IRDY#, когда данные готовы независимо от реакции целевого устройства. Поэтому ошибка четности данных может произойти и на шаге запроса мастера, и на шаге выполнения транзакции мастером. Кроме того, ошибка четности данных может быть константой (т. е. одинаковая ошибка происходит каждый раз, когда мастер повторяет транзакцию) или случайной величиной (т. е. ошибка происходит на некоторых, но не на всех, повторных транзакциях мастера). Методы сообщения об ошибке четности данных для задержанных транзакций записи, описанные в следующих разделах, предназначены, чтобы обнаруживать и сообщать как о постоянных, так и о случайных ошибках четности данных, и предотвращать происхождение состояний блокировки вследствие случайных ошибок четности данных.

Если цель обнаруживает ошибку четности данных при транзакции записи, которая должна была быть обработана как задержанная транзакция, то цель должна выполнить следующую последовательность действий:

1. Выполнить фазу данных, в которой произошла ошибка, выставляя сигнал TRDY#. Если мастер делает попытку блочной передачи, то цель должна также выставить сигнал STOP#.
2. Сообщить об ошибке по линии PERR#.
3. Отменить транзакцию. Задержанный запрос записи не будет установлен в очередь и не останется задержанного выполнения записи.

Если цель обнаруживает ошибку четности данных во время начальной фазы запроса задержанной транзакции записи, задержанный запрос не будет поставлен в очередь. При отсутствии ошибок задержанный запрос записи ставится в очередь. Но цель может обнаружить ошибку четности данных во время следующего повторения транзакции. В этом случае цель не отменяет задержанное выполнение записи, даже если транзакция соответствует предварительно установленному в очередь запросу. Это объясняется тем, что невозможно определить, соответствует ли транзакция предварительно установленной в очередь из-за наличия ошибки. Такая отмена может привести к "висячей строке" задержанного выполнения записи, потому что мастер полагает,

что транзакция выполнена, а цель ожидает повторения первоначального запроса без ошибок. Пустая строка отменяется по истечении таймера отмены цели. Некоторые устройства будут не в состоянии принять другую задержанную транзакцию во время ожидания таймера отмены. Это связано с тем, что цель не должна обрабатывать одновременно несколько задержанных транзакций. Поскольку это состояние является временным, то устройство не будет заблокировано. В этом состоянии устройство обязано выполнять транзакции, которые используют команды записи памяти *Memory Write and Invalidate* и *Memory Write*. [9]

Восстановление ошибок

Реакция системы на установку сигнала SERR# определяется разработчиками. Устройство выставляет сигнал SERR# при обнаружении неисправимой ошибки. Возможным вариантом реакции системы является останов выполнения при отсутствии достаточного количества информации для исправления или восстановления последствий ошибки.

Сигналы для сообщения об ошибках четности на шине PCI и биты состояния разработаны для реализации методов обнаружения и сообщения об ошибках четности данных. Во время транзакций записи цель всегда сообщает об ошибках четности данных мастеру через линию PERR#. Во время транзакции чтения мастер выставляет сигнал PERR#, для указания системе, что была обнаружена ошибка. В обоих случаях мастер имеет возможность передать ошибку его драйверу устройства или операционной системе. Мастер также может пытаться восстанавливать состояние, используя аппаратные и/или программные методы. При обнаружении центральным ресурсом установленного сигнала PERR# может выполняться сообщение об ошибках четности данных операционной системе путем установки сигнала SERR#. При реализации данной необязательной функции восстановление состояния системы невозможно [9].

Примеры исправления ошибок

Восстановление ошибки четности данных является необязательным для мастеров PCI и систем. Следующие примеры описывают варианты восстановления ошибки четности данных мастером, драйвером и операционной системой.

Исправление мастером. Мастер может повторить транзакцию, в которой произошла ошибка четности данных, при наличии информации о возможности повторения транзакции без побочных эффектов. Если при повторе транзакции ошибка не происходит, то сообщать об ошибке четности (операцион-

ной системе или драйверу устройства) не требуется. Если же ошибка повторяется, или если мастер не в состоянии исправить ошибку четности данных, то мастер должен сообщить о ней драйверу устройства. Это может быть реализовано посредством прерывания, либо изменением регистра состояния, установкой флага, или другим соответствующим способом. Если для мастера не существует драйвера устройства, он может сообщить об ошибке установкой сигнала SERR#.

Большинство устройств имеют побочные эффекты при доступе, и поэтому маловероятно, что повторение транзакции даст необходимый эффект. Но в приложениях, где мастер располагает информацией о поведении цели, повторение транзакции может устранить ошибку.

Исправление драйвером устройства. Драйвер устройства может поддерживать исправление ошибок четности данных. В этом случае сообщать операционной системе об ошибке не требуется. Например, драйвер может повторить полную поблочную пересылку путем перезагрузки мастера со всеми исходными данными: размером передачи, источником и адресами назначения данных. Если ошибка не происходит на повторной поблочной пересылке, то об ошибке не сообщается. Драйвер устройства должен сообщить об ошибке операционной системе при отсутствии информации о том, что доступ может быть повторен без побочных эффектов.

Исправление операционной системой. Как только об ошибке четности данных сообщено операционной системе, ни один агент или механизм не должен исправлять ошибку. Методы и способы обработки ошибок на этом уровне определяются самой операционной системой (ее разработчиками) [9].

Задержки на шине

Реализация высокоскоростных интерфейсов, продуманная архитектура и применение новейшей элементной базы еще не гарантируют высокую производительность. Согласованность функционирования устройств, оптимальные алгоритмы и расчет временных задержек — это один из путей к максимально эффективному использованию аппаратной части при организации взаимодействия с устройствами.

Цели и мастера должны применять состояния ожидания как можно реже. Максимальное время пребывания мастера на шине ограничивается программируемым таймером. С учетом этих двух ограничений может быть рассчитано время наибольших задержек на шине PCI для любого мастера. Даже мосты к стандартным шинам расширения с большим временем доступа (ISA, EISA, или MCA) могут быть разработаны таким образом, чтобы оказывать минимальное воздействие на шину PCI [9, 18].

Задержки цели

Под задержкой цели понимается число тактов, которые цель пропускает перед установкой сигнала TRDY#. Требования на начальной фазе данных отличаются от таковых для последующих фаз данных.

Начальные задержки цели

Начальная задержка цели — это число тактов от установки сигнала FRAME# до выставления сигнала TRDY# при выполнении начальной фазы данных, или до выставления сигнала STOP# при завершениях с типом "Target-Abort" и "Retry". Число пропускаемых тактов изменяется в зависимости от типа команды чтения или записи и, для команды записи, от возможности буферизации. Команда записи в память должна быть сохранена целью в буфере и записана по адресу предназначения позже. В этом случае начальная задержка цели является малой величиной, потому что транзакция, по сути, была просто передачей из регистра в регистр. Выполнение требований начальной задержки на транзакциях чтения более трудноосуществимо, т. к. в этом случае задержка зависит от времени доступа самих носителей данных (например, HDD, DRAM и т. д.) и задержки логики интерфейса. Выполнение требований начальной задержки для операций ввода-вывода и транзакций записи конфигурации сопоставимо с задержками при чтении. Требования начальной задержки для цели зависят от действий, выполняемых системой. Система может функционировать в режиме инициализации или в режиме выполнения. Режим инициализации начинается по снятию сигнала RST# и по прошествии 2^{25} тактов на шине PCI. Режим выполнения следует за режимом инициализации.

Если доступ к цели осуществляется в режиме инициализации, то ей разрешено выполнить одно из следующих действий:

- ❒ игнорировать запрос (если цель не загрузочное устройство);
- ❒ разрешить доступ и находиться в состоянии ожидания, пока она не сможет выполнить запрос, не выходя за рамки режима инициализации;
- ❒ разрешить доступ и закончить его с типом завершения "Retry".

Если доступ к цели осуществляется в режиме выполнения (сигнал RST# был снят больше чем 2^{25} тактов), то она должна выполнить начальную фазу данных транзакции (чтения или записи) в пределах 16 тактов от установки сигнала FRAME#. Цель выполняет начальную фазу данных, выставляя сигнал TRDY# (чтобы принять или предоставить требуемые данные), или заканчивает запрос, выставляя сигнал STOP# в пределах требований начальной задержки. Главным мостам шины для выполнения начальной фазы данных предоставлены дополнительные 16 тактов, если доступ производится в мо-

дифицированную строку кэша. Максимальное ограничение для главного моста составляет 32 такта на любой начальной фазе данных. На практике начальная задержка фазы данных становится известна, когда устройство уже разработано. Если время, необходимое для выполнения начальной фазы данных, будет превышать значение максимальной начальной задержки, устройство должно завершить транзакцию с типом "Retry" как можно быстрее и выполнить транзакцию как задержанную. В том случае, если начальная задержка фазы данных не может быть определена заранее, то допускается реализация счетчика. Этот счетчик должен заставить цель выставить сигнал STOP# и начать выполнение транзакции как задержанной на шестнадцатом такте или раньше, если не выставлен сигнал TRDY#. Цель, которая находится в состоянии ожидания по 16 тактов каждый раз до начала задержанной транзакции, может снижать производительность. Поэтому подобная реализация на основе счетчика является крайне нежелательной.

Все цели должны выполнять требование 16 тактов начальной задержки, как описано ранее. Новые устройства не должны зависеть от целей, выполняющих требование максимума начальной задержки 16 тактов. Тем не менее устройствам придется функционировать в обычном режиме (хотя с уменьшенной производительностью), т. к. системы и устройства, разработанные для ранних версий спецификации шины PCI, не могут выполнить новые требования. Новые устройства должны быть обратно совместимыми.

Целям даются три варианта для выполнения требования начальной задержки. Большинство целей будет использовать алгоритм 1 или алгоритм 2. Устройства, неспособные использовать алгоритм 1 или 2, должны использовать алгоритм 3.

Алгоритм 1. Алгоритм относится к устройству, которое всегда передает данные (выставляет сигнал TRDY#) в пределах 16 тактов от момента установки сигнала FRAME#. Большинство контроллеров ввода-вывода, разработанных ранее, могут выполнить требования начальной задержки алгоритма 1. Цель всегда выставляет сигнал TRDY# для выполнения начальной фазы данных транзакции в пределах 16 тактов от установки сигнала FRAME#.

Алгоритм 2. Этот алгоритм предназначен для устройств, которые обычно передают данные в пределах 16 тактов, но в некоторых определенных состояниях могут превышать пределы начальной задержки. В таких состояниях устройство завершает доступ с типом "Retry" в пределах 16 тактов от установки сигнала FRAME#. Для устройств, которые не могут использовать алгоритм 1, может потребоваться небольшая модификация для выполнения условий алгоритма 2. В случае завершения с типом "Retry" мастер должен повторить транзакцию. Цели разрешено функционировать по такому алгоритму в предположении, что цель будет способна выполнить повторную транзакцию. В противном случае цель должна использовать алгоритм 3.

В качестве примера рассмотрим простое графическое устройство, которое обычно отвечает на запрос в пределах 16 тактов, но в определенных состояниях, типа регенерации изображения, внутренняя шина занята и препятствует передаче данных. В этом состоянии цель заканчивает доступ с типом "Retry", т. к. при повторении транзакции мастером цель будет в состоянии выполнить передачу. Устройство может иметь внутренний сигнал, который указывает интерфейсу шины, что внутренняя шина занята, и данные в это время не могут передаваться. Это позволяет устройству требовать доступа (выставляя сигнал DEVSEL#) и немедленно завершать доступ с типом "Retry". Применение такого сигнала вместо того, чтобы ждать 16 тактов после выставления сигнала FRAME#, позволяет другим агентам использовать шину.

Алгоритм 3. Алгоритм предназначен для устройства, которое обычно не в состоянии передавать данные в пределах 16 тактов. Эта опция требует, чтобы устройство использовало задержанные транзакции.

Устройства, которые не в состоянии выполнить требования алгоритма 1 или 2, обязаны использовать алгоритм 3. Эта опция используется устройствами, которые при нормальных условиях не могут закончить транзакцию в пределах требований начальной задержки. Примером может служить контроллер ввода-вывода, который имеет несколько внутренних функций, конкурирующих за интерфейс PCI для получения доступа к внутреннему ресурсу. Иным примером может быть устройство, которое взаимодействует подобно мосту с другим устройством или шиной, где начальная задержка может быть больше чем 16 тактов.

Комбинация различных алгоритмов может использоваться, базируясь на задержке доступа специфического устройства. Например, графический контроллер может выполнить требования начальной задержки, используя алгоритм 1 для конфигурационного доступа или доступа к внутренним регистрам. При доступе к буферу кадров он должен будет использовать алгоритм 2 или, в некоторых случаях, алгоритм 3 [9, 18].

Задержки в блоках

Под задержкой в блоке понимается количество тактов между установкой сигналов IRDY# и TRDY# для одной фазы данных и установкой сигнала TRDY# или STOP# для следующей фазы данных в блоке. Цель обязана выполнять следующую фазу данных в пределах восьми тактов после завершения предыдущей фазы данных. Это правило требует, чтобы цель выполнила фазу данных любым из доступных способов: передавая данные, выполняя "Disconnect without Data" или "Target-Abort" в пределах восьми тактов. В большинстве случаев задержка между блоками становится известна уже после разработки устройства. Цель должна манипулировать линиями TRDY#

и STOP# так, чтобы транзакция заканчивалась после выполнения фазы данных "N" (где N = 1, 2, 3...), если задержка фазы данных "N+1" будет больше чем восемь тактов. Предположим, что мастеру на шине PCI требуется минимум 15 тактов на каждую фазу данных для чтения из шины расширения. Применяя правило для N = 1, задержка для фазы данных 2 составит 15 тактов; таким образом, цель должна завершить транзакцию после выполнения фазы данных 1, т. е. разорвать блок на границе фазы данных.

Если цель не в состоянии выполнить требование восьми тактов, то допускается реализация счетчика. Это заставит цель выставлять сигнал STOP# до или на восьмом такте, если не выставлен сигнал TRDY#. Если сигнал TRDY# выставлен в пределах восьми тактов, то счетчик сбрасывается и цель продолжает транзакцию [9, 18].

Задержка мастера

Задержка мастера — это число тактов, которые требуются мастеру для установки сигнала IRDY# (IRDY# указывает на готовность передачи данных). Все мастера обязаны выставлять сигнал IRDY# в пределах восьми тактов от момента установки сигнала FRAME# на начальной фазе данных и в пределах восьми тактов на всех последующих фазах данных. В первой фазе данных транзакции мастер не должен задерживать установку сигнала IRDY# больше, чем на один или два такта для транзакции записи. Мастер не должен задерживать установку сигнала IRDY# на транзакции чтения. Если мастер не имеет доступного буфера для хранения читаемых данных, то он должен задержать запрос на использование шины, пока не будет доступен буфер. Во время транзакции записи мастер должен иметь данные в наличии перед тем запросом на передачу. Для повышения производительности пересылки данных по шине PCI должны быть реализованы как межрегистровый обмен [9].

Максимальное время записи в память

Цель может завершать транзакцию записи в память с типом "Retry", пока решаются временные внутренние конфликты; например, когда переполнены все буферы данных для записи в память, или во время регенерации изображения. При этом постоянно завершать транзакцию записи в память с типом "Retry" цель не должна. После того как цель завершает транзакцию записи в память с типом "Retry", она должна быть в состоянии выполнить, по крайней мере, одну фазу данных записи в память в пределах указанного числа PCI-тактов от первого завершения "Retry". Это число циклов тактов составляет 334 такта для систем, работающих на частоте 33 МГц, и 668 тактов для систем, работающих на частоте 66 МГц. Данный временной интервал, который равен 10 мкс на максимальных частотах (33 и 66 МГц), называется *максимальным*

временем выполнения. Если к цели осуществляется несколько запросов записей в память, то максимальное время выполнения измеряется от момента завершения первой транзакции записи в память с типом "Retry" до завершения первой фазы данных любой записи в память, отличного от "Retry". После любого другого, отличного от "Retry", завершения, максимальное время выполнения обнуляется. Фактическое время выполнения фазы данных будет также зависеть от того, когда мастер повторяет транзакцию. Устройства должны разрабатываться исходя из условия, что мастер повторит транзакцию записи в память в пределах максимального времени выполнения [9].

Предел максимального времени выполнения

Некоторые устройства не смогут обработать каждую транзакцию записи в память в пределах максимального времени выполнения. Примером является запись в очередь команд, где команды для выполнения могут потребовать больше времени, чем максимальное время выполнения. Совокупность записей к такой цели может привести к более длительным временным задержкам, чем допускается. Такие устройства должны быть ограничены в использовании, что может быть обеспечено драйвером устройства. Для этого ограничивается частота, с которой осуществляются записи в память, либо выполняется чтение состояния устройства.

Требования максимального времени выполнения не предъявляются к мостам. Мосты должны обеспечивать требованиям максимального времени выполнения для транзакций, непосредственно адресованным к мостам.

Максимальное время выполнения также не относится к инициализации устройства, которое определено как 225 тактов шины PCI после снятия сигнала RST#.

Несмотря на обязательное требование, фактическое время выполнения возможно превысит установленный предел. Это объясняется тем, что транзакция может проходить через мост PCI-to-PCI или быть одной из подряд идущих транзакций к цели.

Арбитражные задержки

Задержка арбитража — это количество тактов от момента установки мастером сигнала REQ# до перехода шины в состояние "свободно" и установки им сигнала GNT#. В ненагруженной системе арбитражная задержка представляет собой время, за которое арбитр шины выставляет сигнал мастера GNT#. Если во время установки мастером сигнала GNT# выполняется другая транзакция, то мастер должен ждать дополнительное время для завершения текущей транзакции. Полная арбитражная задержка представляет собой совокуп-

ность времени предоставления шины другим мастерам и времени удержания шины каждым из них. Количество и порядок предоставления мастерам шины определяется арбитром шины. Время удержания шины мастером ограничено таймером задержки мастера после снятия сигнала GNT#. Величина таймера задержки мастера — это программируемое значение в пространстве конфигурации каждого мастера. Таймер требуется каждому мастеру, который способен к передаче блоками больше чем двух фаз данных. Таймер задержки каждого мастера сбрасывается и останавливается при снятии сигнала FRAME#. Установка мастером сигнала FRAME# запускает таймер. Поведение мастера по истечении таймера зависит от используемой команды и состояния сигналов FRAME# и GNT#.

- Если мастер снимает сигнал FRAME# раньше или одновременно с истечением времени таймера, то происходит обычное завершение.
- Если истекло время таймера, а сигнал FRAME# находится в активном состоянии, и при этом командой шины является любая, кроме *Memory Write and Invalidate*, то мастер должен начать завершение транзакции по снятию сигнала GNT#. В этом случае мастер указал цели, что он выполнит текущую фазу данных и следующую за ней (заключительная фаза данных указывается снятием сигнала FRAME#).
- Если истекло время таймера, а сигнал FRAME# находится в активном состоянии, и при этом командой является *Memory Write and Invalidate*, то последовательность меняется. При условии, что текущая фаза данных не передает последнее двойное слово текущей строки кэша, мастер должен завершить транзакцию по снятию сигала GNT# в конце текущей строки кэша (или когда выставлен сигнал STOP#).
- Такая же ситуация: истекло время таймера, сигнал FRAME# находится в активном состоянии, и при этом командой является *Memory Write and Invalidate*. Но если текущая фаза данных передает последнее двойное слово текущей строки кэша, то мастер должен завершить транзакцию по снятию сигнала GNT# в конце следующей строки кэша. Данное требование относится к мастеру, выполнившему, по крайней мере, одну фазу данных, которая является началом следующей строки кэша и которую он должен закончить. Он не должен выполнять это требование, только если выставлен сигнал STOP#.

По сути, величина таймера задержки представляет собой минимум количества тактов, выделенных мастеру, по прошествии которых он должен освободить шину в максимально короткий срок после снятия сигнала GNT#. Фактическая продолжительность транзакции (в предположении, что вовремя снят сигнал GNT#) может составлять величину от минимума значения таймера задержки плюс один такт до максимума значения таймера задержки плюс

число тактов, требуемое для завершения передачи строки кэша (при условии, что цель не снимает сигнал STOP#) [9].

Влияние задержек на скорость передачи

В PCI-системах существует компромисс между малым временем задержки и шириной полосы пропускания, а следовательно, производительности. Высокая производительность достигается путем разрешения устройствам использовать блочные передачи. Малое время задержки достигается путем уменьшения максимального размера блочной передачи. В этих условиях необходимо оптимальное решение, либо сознательное изменение равновесия для достижения определенных целей.

Данный мастер шины инициирует задержку на PCI при каждом использовании шины для выполнения транзакции. Эта задержка представляет собой функцию поведения мастера и цели во время транзакции и состояния сигнала мастера GNT#. Используемая команда шины, длина блока транзакции, задержка данных мастера для каждой фазы данных и значение таймера задержки являются основными параметрами, которые управляют поведением мастера. Используемая команда шины, задержка цели и задержки в блоках являются основными параметрами, которые управляют поведением цели. Мастер обязан выставлять свой сигнал IRDY# в пределах восьми тактов для любой фазы данных (начальной и последующей). Для первой фазы данных цель обязана выставить сигналы TRDY# или STOP# в пределах 16 тактов от момента установки сигнала FRAME# (если доступ не попадает на модифицированную строку кэша), причем для главных мостов шины допускается 32 такта. Для всех следующих после первой фаз данных в блоке цель должна выставлять ее сигналы TRDY# или STOP# в пределах восьми тактов. Если не учитывать таймер задержки, то для расчета худшего случая задержек, которые может ввести мастер шины PCI, могут использоваться следующие уравнения [9]:

latency_max = 32 + 8 × (*n* – 1), если доступ попал в модифицированную строку кэша;

latency_max = 16 + 8 × (*n* – 1), при обычных условиях,

где *n* — общее количество передач данных в транзакции, а latency_max — максимальная задержка в тактах. Первое уравнение относится к главным мостам шины, второе ко всем другим устройствам. Более важным является рассмотрение транзакций, которые показывают обычное поведение. Шина PCI разработана так, чтобы передачи данных между мастером шины и целью происходили аналогично межрегистровым передачам. Поэтому мастера шины обычно не вставляют состояния ожидания, т. к. они только тогда запрашивают транзакции, когда готовы передать данные. Стандартные цели имеют начальное время ожидания доступа меньше чем 16 максимально разрешен-

ных тактов. Как только цели начинают передавать данные (выполнять их первую фазу данных), они, как правило, в состоянии обработать блочные передачи на полной скорости (один такт на фазу данных). Цель может использовать протокол завершения типа "Disconnect", чтобы закончить блочную транзакцию раньше. С учетом более реальных условий уравнения задержки для худшего случая могут быть изменены, чтобы получить типичное время ожидания. В качестве начальных условий рассматривается начальная задержка цели восемь тактов, а таймер задержки игнорируется.

latency_typical = 8 + (*n* – 1),

где latency_typical — типичная задержка в тактах.

Если мастерам будет разрешена блочная передача с целью, которая могла бы принимать и передавать данные с неопределенными задержками, то значение максимальной задержки будет также неопределенно. Как раз для предотвращения неопределенных задержек применяется таймер задержки мастера. Таймер обеспечивает механизм, чтобы ограничить время пребывания мастера на шине. В действительности, таймер задержки именно и позволяет регулировать соотношение производительности и величины задержки. В табл. 6.2 данные соотношения приведены с учетом следующих предположений:

- начальная задержка мастера или цели составляет восемь тактов;
- между последовательными фазами данных задержка отсутствует;
- количество фаз данных представляет степень двух, потому что они коррелируют с размерами строки кэш-памяти;
- значения таймера задержки выбираются таким образом, чтобы счетчик достигал предельной величины во время, приблизительно совпадающее с последней фазой данных. Это позволяет мастеру выполнять корректное количество фаз данных.

Например, для значения таймера задержки 14 тактов и начальной задержки цели 8 таймер истекает во время седьмой фазы данных и транзакция завершается после восьмой фазы данных.

Таблица 6.2. *Соотношение задержки и размера передачи*

Кол-во фаз данных	Передано байт	Всего тактов	Таймер задержки (тактов)	Скорость (Мбайт/с)	Задержка (мкс)
8	32	16	14	60	0,48
16	64	24	22	80	0,72
32	128	40	38	96	1,20
64	256	72	70	107	2,16

В таблице представлены следующие параметры:

- **Кол-во фаз данных** — количество фаз данных, выполненных в транзакции;
- **Передано байт** — полная сумма переданных в транзакции байт (32-битный обмен);
- **Всего тактов** — количество тактов для выполнения транзакции, может быть рассчитано по следующей формуле:

  total_clocks = 8 + (*n* – 1) + 1,

  где один добавленный такт предназначен для состояния шины "свободно";
- **Таймер задержки** — значение таймера задержки в тактах. Таймер истекает на такте, следующем после заключительной фазы данных. Может быть рассчитано по следующей формуле:

  latency_timer = total_clocks – 2
- **Скорость** — рассчитанная скорость передачи в мегабайтах в секунду. Определяется по формуле:

  скорость_передачи = число байт/(число тактов × 30 нс)
- **Задержка** — задержка в микросекундах, потребовавшаяся транзакции. Можно рассчитать по формуле:

  задержка = число тактов × 30 нс

Из рассмотрения табл. 6.2 видно, как увеличение длины блока влияет на увеличение количества переданных данных. Количество переданных данных удваивается, в то время как задержка увеличивается меньше чем в два раза. Количество переданных данных первого ряда по сравнению с последним увеличилось в 8 раз, в то время как задержка увеличивается в 4,5 раза. Чем длиннее транзакция (больше фаз данных), тем более эффективно используется шина. Такое увеличение производительности может быть реализовано за счет буферов большего объема [9].

Расчет задержки арбитража

Задержка арбитража зависит от алгоритма арбитража системы; т. е. от последовательности, в которой мастерам предоставляется доступ к шине и величины таймера задержки каждого мастера. Эти параметры могут изменяться для различных реализаций. Следовательно, при реализации мастера рекомендуется исходить из типичного значения задержки. Задержка арбитража также влияет на нагрузку системы и эффективность использования шины. В следующих двух примерах рассмотрены различные варианты загруженности системы. Для 32-битной шины PCI рассмотрены ненагруженная и перегру-

женная системы. Оба примера являются прототипами существующих систем [9].

Ненагруженная система. Для данного примера предполагается, что не выставлены сигналы REQ#, и шина находится либо в состоянии "свободно", либо мастер в настоящее время использует шину. Поскольку не выставлен ни один из сигналов REQ#, то как только сигнал REQ# агента A выставлен, арбитр установит его сигнал GNT# при следующем определении линий REQ#. В этом случае сигнал GNT# агента A будет выставлен в пределах нескольких тактов. Агент A получает доступ к шине, когда шина находится в состоянии "свободно" (в предположении, что его сигнал GNT# является все еще активным).

Перегруженная система. В данном рассмотрении используется пример, описанный в *главе 5*. Предположим, что сигналы REQ# всех агентов находятся в активном состоянии, и все агенты собираются передать больше данных, чем позволяют их таймеры задержки. Пусть значение таймера задержки составляет 22 такта. Чтобы начать предоставлять шину агентам, предположим, что следующий мастер шины — агент A на уровне 1 и агент X на уровне 2. Агент A имеет наименьшее число тактов задержки до получения доступа к шине, Агент Z имеет наибольшее число. Обозначим за оборот цикл от запроса мастера доступа до освобождения шины и готовности к арбитражу. В этом примере, агенты A и B каждый совершает оборот перед агентом на уровне 2. Поэтому агенты A и B совершают три оборота на шине, и агенты X и Y совершают один оборот прежде, чем агент Z совершает оборот. Арбитражная задержка может быть короткой, как несколько тактов для агента A, или длинной, как 176 тактов (8 мастеров × 22 такт/мастер) для агента Z. Для перегруженной системы может потребоваться более высокая скорость передачи данных, около 90 Мбайт/с, принимая начальную задержку цели восемь тактов и задержку в блоках один такт).

Как показано ранее, максимальная арбитражная задержка для отдельно взятого мастера достигается, когда все другие мастера используют шину в пределах значений их таймеров задержки. Вероятность увеличения задержки зависит от увеличения нагрузки шины. В ненагруженной системе меньшее количество мастеров будут требовать доступа к шине. Либо они будут использовать шину, не выходя за пределы таймера задержки, таким образом обеспечивая более быстрый доступ другим мастерам.

На эффективность использования шины каждым агентом будут также влиять средние арбитражные задержки. Чем больше состояние ожидания, которые мастер или цель вставляют на каждой транзакции, тем дольше каждая транзакция будет продолжаться, таким образом, увеличивая вероятность, что каждый мастер будет использовать шину вплоть до максимума таймера задержки.

В следующих примерах рассмотрено воздействие на арбитражные задержки, поскольку эффективность шины понижается вследствие вставляемых состояний ожидания. В обоих примерах система имеет единственный арбитражный уровень, значение таймера задержки составляет 22 такта и присутствуют пять мастеров, которым необходимо передать данные. Значение таймера задержки в 22 такта позволяет каждому мастеру передавать 64 байта строки кэш-памяти, при условии, что начальная задержка составляет восемь тактов, и задержка в блоке составляет один такт. Пример эффективной системы показывает снижение арбитражной задержки при оптимальном использовании шины. В примере с неэффективным использованием шины показано увеличение арбитражной задержки в результате нерационального использования шины.

Система с эффективной шиной. В этом примере каждый мастер в состоянии передать всю 64-байтную строку кэша до истечения таймера задержки. Предполагается, что каждый мастер готов передать другую строку кэш-памяти только после выполнения текущей транзакции. Таймер задержки исключается из рассмотрения. Для передачи каждой строки кэша мастеру потребуется:

[(1 такт "свободно") + (8 начальных тактов TRDY#) + (15 следующих тактов TRDY#)] × 30 нс/такт = 720 нс.

Если все пять мастеров используют такое же число тактов, то каждый мастер должен будет ждать четырех других:

720 нс × 4 = 2880 нс = 2,9 мкс.

Таким образом, период доступа каждого мастера на шине составляет 2,9 мкс. Каждый мастер передает данные со скоростью приблизительно 90 Мбайт/с.

Система с неэффективной шиной. Рассмотрим систему, в которой присутствуют два существенных недостатка. Первый недостаток состоит в том, что истекает таймер задержки. Вторым недостатком являются две транзакции, предназначенные для выполнения передачи одной строки кэш-памяти, что приводит к увеличению загрузки системы. В этом примере начальная задержка составляет 8 тактов, а задержка в блоке составляет 2 такта. Таймер задержки истечет прежде, чем мастер передаст всю 64-байтную строку кэша. По истечении таймера задержки снимается сигнал GNT#, а сигнал FRAME# выставлен и мастер должен остановить транзакцию преждевременно. При этом он должен выполнить две заключительные фазы данных, которые он начал выполнять (для команды *Memory Write and Invalidate* это текущая строка кэш-памяти). Срок пребывания каждого мастера на шине был бы таким:

[(1 такт "свободно") + (22 такта таймера задержки) + (2 × 2 такта TRDY#)] × 30 нс/такт = 810 нс.

Каждый мастер должен будет ждать четырех других:

810 нс × 4 = 3240 нс = 3,2 мкс.

Следовательно, период доступа каждого мастера на шине составляет 3,2 мкс. Таким образом, каждому мастеру потребовалось немногим больше времени, чем в предыдущем примере. Но при этом мастер выполнил только 9 фаз данных (36 байтов, чуть больше половины строки кэш-памяти) вместо 16 фаз данных. Каждый мастер передает данные со скоростью приблизительно 44 Мбайт/с.

Арбитражная задержка в примере с неэффективной шиной составляет 3,2 мкс по сравнению с 2,9 мкс для примера эффективной шины. Но во втором примере для передачи одной строки кэша мастеру требуется две транзакции. Это удвоило загрузку системы, не увеличивая производительность. Такой результат получился из-за добавления одного состояния ожидания к каждой фазе данных.

В примерах предполагается, что все пять мастеров находятся на одном арбитражном уровне. Если мастер находится на более низком арбитражном уровне или располагается с другой стороны моста PCI-to-PCI, то его период доступа на шину увеличивается.

Максимальные пределы задержек предназначены для мгновенных состояний, в то время для нормального поведения рекомендуются средние значения. Примером мгновенного состояния может служить неспособность устройства продолжать фазу данных на каждом такте. Вместо того чтобы остановить передачу, цель вставила бы несколько состояний ожидания и продолжила блок, выполняя фазу данных на каждом такте. Максимальные пределы не предназначены для использования в каждой фазе данных, они используются в тех редких случаях, когда данные временно не могут быть переданы [9].

Определение буферных требований

Скорость передачи данных зависит от медленных устройств. Необходимо минимизировать разницу между скоростью, с которой устройство принимает или передает данные, и скоростью, с которой данные могут быть переданы по шине. Для этого применяется метод буферизации (иначе говоря, метод примитивного кэширования). Объем буферизации может быть определен несколькими факторами, основанными на функциональных возможностях устройства и его скорости, с которой оно обрабатывает данные. Арбитражная задержка мастера и эффективность передачи данных на шине будут влиять на объем буферизации, который требуется устройству. В некоторых случаях не требуется буферизации большого размера, например для обработки ошибок, в то время как буферизация большего объема может дать лучшее использование шины. Для устройств, не использующих шину интенсивно (таким уст-

ройствам редко требуется больше чем 5 Мбайт/с), рекомендуется размер буферов равный минимум 4 двойных слова. Это позволит гарантировать, что транзакции на шине выполняются оптимально. Предпочтительный размер передачи данных составляет полную строку кэша. Транзакции, в которых передается меньше чем 4 двойных слова, являются неэффективными и зря занимают шину. Устройствам, в большей степени использующим шину (таким устройствам требуется больше чем 5 Мбайт/с), рекомендуется поддерживать размер буферизации минимум 32 двойных слова. Это позволит гарантировать оптимальное выполнение транзакций. Устройства, использующие шину неэффективно, будут отрицательно влиять на работу системы в целом и оказывать еще большее воздействие на будущие разработки. Приведенные рекомендации являются минимальными, реальный же объем буферизации прямо пропорционален максимально возможным перегрузкам или недогрузкам. Например, контроллер диска обеспечил достаточную буферизацию, чтобы передать данные эффективно через шину PCI, но не обеспечил никакой дополнительной буферизации для недогрузок и перегрузок (в предположении, что они не будут происходить). Если запись на диск недоступна, то контроллер будет ждать разрешения доступа. Для операций чтения контроллер не будет принимать новых данных, т. к. недоступен его буфер.

Плата захвата изображения должна сбрасывать буферы прежде, чем поступают новые данные, иначе данные будут повреждены. Для качественной реализации видеоподсистемы необходимо обеспечить способ для этого агента, позволяющий предоставлять ему достаточную полосу пропускания шины. Это может быть достигнуто за счет реализации арбитра с различными уровнями приоритета и/или манипуляцией таймером задержки других мастеров, чтобы ограничить срок их пребывания на шине.

Последующие разработки должны включать устройства, использующие шину настолько эффективно, насколько это возможно, т. е. передавать такие большие объемы данных, насколько возможно (предпочтительно несколько строк кэша) за наименьшее число тактов. Поскольку такая передача должна быть реализована всеми устройствами, то система будет работать более производительно и с меньшими задержками. Из требования уменьшения задержек следует, хотя и необязательное, уменьшение размеров буферов устройств. Кроме того, для определения эффективности использования шины требуются специальные программные и аппаратные тесты [9].

Обобщение правил функционирования

В данном разделе все правила взаимодействия устройств, управления служебными и другими линиями, описание правил перехода шины из состояния в состояние и основы протокола PCI сведены в итоговый перечень [9].

Условия стабильности сигналов

1. Следующие сигналы будут стабильными на всех передних фронтах сигнала CLK после сброса: LOCK#, IRDY#, TRDY#, FRAME#, DEVSEL#, STOP#, REQ#, GNT#, REQ64#, ACK64#, SERR# (только по заднему фронту) и PERR#.
2. Правила для состояния линий адреса/данных следующие:
   - сигналы по линии адресов AD[31::00] стабильны на первом такте, на котором зафиксировано установка сигнала FRAME#;
   - сигналы по линии адресов AD[63::32] стабильны и действительны во время первого такта после установки сигнала REQ64#, когда используется 32-битная адресация (SAC), или первые два такта после установки REQ64#, когда используется 64-битная адресация (DAC). Когда сигнал REQ64# снят, то сигналы AD[63::32] должны "подтягиваться" к необходимому уровню центральным ресурсом;
   - сигналы линий данных AD[31::00] стабильны и действительны независимо от того, какие из них участвуют в транзакции чтения при установке сигнала TRDY# и в транзакции записи при установке сигнала IRDY#. В любое другое время эти линии могут находиться в плавающем состоянии. Состояние линий AD не может изменяться до конца текущей фазы данных после установки сигнала IRDY# на транзакции записи или сигнала TRDY# на транзакции чтения;
   - сигналы линий данных AD[63::32] стабильны и действительны независимо от того, какие из них участвуют в транзакции чтения при установке сигналов ACK64# и TRDY# и в транзакции записи при установке сигнала IRDY#. В любое другое время эти линии могут находиться в плавающем состоянии;
   - сигналы линий данных AD[31::00] при выполнении команды специального цикла стабильны и действительны независимо от того, какие из них участвуют в транзакции при установке сигнала IRDY#;
   - асинхронные данные по шине PCI не передаются, пока в транзакции записи выставлен сигнал IRDY# и в транзакции чтения выставлен сигнал TRDY#.
3. Правила для состояния линий C/BE# — команда/разрешение обращения к байтам следующие:
   - сигналы линий C/BE[3::0]# являются стабильными и действительными для первой установки сигнала FRAME# и содержат коды команд. Линии C/BE[7::4]# должны быть стабильными и действительными во время первого такта после установки сигнала REQ64# при использовании

32-битной адресации (SAC) и являются зарезервированными. Линии C/BE[7::4]# должны быть стабильными и действительными во время первых двух тактов после установки сигнала REQ64# при использовании 64-битной адресации (DAC) и содержат фактическую команду шины. Когда сигнал REQ64# находится в неактивном состоянии, линии C/BE[7::4]# "подтягиваются" до стабильного состояния центральным ресурсом;

- сигналы линий разрешения обращения к байтам C/BE[3::0]# должны быть стабильными и действительными один такт после фазы адреса и каждой законченной фазы данных, а также оставаться действительными каждый такт во время всей фазы данных независимо от состояний ожидания. Эти линии указывают, какие линии данных содержат действительные данные. Линии C/BE[7::4]# функционируют так же, как и линии C/BE[3::0]#, за исключением активного состояния сигнала REQ64#, когда они охватывают старшие 4 байта.

4. Сигнал PAR стабилен и устойчив один такт после установки сигналов по линиям AD[31::00]. Сигнал PAR64 стабилен и действителен один такт после установки сигналов на линиях AD[63::32].
5. Сигнал IDSEL стабилен и действителен только во время первого такта установки сигнала FRAME#, когда выполняется команда конфигурации. В любое другое время состояние линии IDSEL не определенно.
6. Сигналы RST#, INTA#, INTB#, INTC# и INTD# являются асинхронными.

Управление сигналами мастера

1. Транзакция начинается после первой установки сигнала FRAME#.
2. Управление сигналами FRAME## и IRDY# на всех PCI-транзакциях осуществляется в соответствии со следующими правилами:

   - состояние сигналов FRAME# и IRDY# определяет состояние шины "занято" или "свободно"; когда любой из них выставлен, шина занята; когда оба сигнала сняты, шина находится в состоянии "свободно";
   - после перехода в неактивное состояние сигнал FRAME# не может быть выставлен в той же транзакции;
   - сигнал FRAME# не может быть снят, если IRDY# находится в неактивном состоянии. Сигнал IRDY# должен всегда выставляться на первом уровне такта, на котором снимается сигнал FRAME#;
   - после установки мастером сигнала IRDY# он не может изменить состояние сигнала IRDY# или FRAME# до конца текущей фазы данных;

- мастер должен снять сигнал IRDY# на следующем такте после выполнения последней фазы данных.

3. Транзакция завершена по снятию сигналов FRAME# и IRDY#.
4. Если транзакция завершается целью (выставлен сигнал STOP#), перед повтором транзакции мастер должен снять с цели сигнал REQ#. Устройство, содержащее одного мастера, должно снимать сигнал REQ# как минимум на два такта; один такт для перехода шины в состояние "свободно" (в транзакции, где был выставлен сигнал STOP#) и один такт до или после. Устройство, содержащее несколько мастеров, может использовать шину без снятия сигнала REQ#, даже если одна или более транзакций завершены целью. Но устройство должно снять сигнал REQ# на два такта, если шина переходит в состояние "свободно".
5. Транзакция мастера, завершенная целью с типом "Retry", должна быть в точности повторена, пока она не выполнится; мастер не обязан повторять транзакцию, завершенную с типом "Disconnect".

Управление сигналами цели

1. Следующие общие правила определяют поведение сигналов FRAME#, IRDY#, TRDY# и STOP# при завершении транзакций:
   - фаза данных выполняется на любом переднем фронте синхроимпульса, на котором выставлены сигналы IRDY# и один из сигналов STOP# или TRDY#;
   - независимо от состояния сигнала STOP#, передача данных выполняется на каждом переднем фронте такта, если выставлены сигналы IRDY# и TRDY#;
   - после установки целью сигнала STOP# он не должен быть снят до снятия сигнала FRAME#;
   - после установки сигналов TRDY# или STOP# цель не может изменить состояние сигналов DEVSEL#, TRDY# или STOP# до конца текущей фазы данных;
   - мастер должен снять сигнал FRAME#, если выставлен сигнал STOP#, как только он сможет выставить сигнал IRDY#;
   - сигналы TRDY#, STOP# и DEVSEL# должны быть сняты на такте после завершения последней фазы данных и должны находиться в третьем состоянии на следующем такте при условии, что они не были сняты раньше.
2. Агент становится целью доступа, выставляя сигнал DEVSEL#.

3. Сигнал DEVSEL# должен быть выставлен на такте или до него, на котором цель включает ее выходы (линии TRDY#, STOP# или при чтении — AD).
4. После выставления сигнал DEVSEL# не может быть снят до выполнения последней фазы данных, кроме завершения с типом "Target-Abort".

Фазы данных

1. Источник данных обязан выставить его сигнал xRDY# безоговорочно, когда данные действительны (сигнал IRDY# в транзакции записи, сигнал TRDY# в транзакции чтения).
2. Данные передаются между мастером и целью на каждом такте, для которого выставлены сигналы IRDY# и TRDY#.
3. Последняя фаза данных считается выполненной, если:
   - сигнал FRAME# снят, сигнал TRDY# выставлен (нормальное завершение);
   - сигнал FRAME# снят, сигнал STOP# выставлен (завершение целью);
   - сигнал FRAME# снят и истек таймер выбора устройства ("Master-Abort");
   - сигнал DEVSEL# снят и сигнал STOP# выставлен ("Target-Abort").
4. Выполнение фазы данных происходит, когда цель выставляет или сигнал TRDY#, или сигнал STOP#. Цель выполняет следующие операции:
   - передача данных в текущей фазе данных и продолжение транзакции (для блока) установкой сигнала TRDY#;
   - передача данных в текущей фазе данных и завершение транзакции установкой сигналов TRDY# и STOP#;
   - завершение транзакции установкой сигнала STOP# и снятием TRDY#. В текущей фазе данных передача данных не происходит;
   - завершение транзакции установкой сигналов STOP# и снятием сигналов TRDY# и DEVSEL#, т. е. завершение с типом "Target-Abort". Данные не передаются.
5. Цель не выполняет текущую фазу данных, если сняты оба сигнала TRDY# и STOP#. В этом случае цель вставляет состояния ожидания.

Алгоритм арбитража

1. Агенту разрешено начать транзакцию только в следующих двух случаях:
   - выставлен сигнал GNT# и шина находится в состоянии "свободно" (сняты сигналы FRAME# и IRDY#);

- сигнал GNT# выставлен в последней фазе данных транзакции, и агент начинает быструю транзакцию back-to-back (снят сигнал FRAME# и выставлен TRDY# или STOP#, или транзакция завершена с типом "Master-Abort").

2. Арбитр может снять сигнал агента GNT# на любом такте.
3. Перевод сигнала GNT# из активного состояния в неактивное выполняется согласно следующим правилам:
   - если сигнал GNT# снят, а FRAME# выставлен на одном и том же такте, то транзакция шины продолжится;
   - снятие одного сигнала GNT# может совпадать с установкой другого сигнала GNT#, если шина не находится в состоянии "свободно". В противном случае требуется задержка в один такт между снятием сигнала GNT# и установкой другого GNT# для предотвращения появления сигналов на линиях AD и PAR;
   - если сигнал FRAME# снят, то сигнал GNT# может быть снят в любое время для обслуживания мастера с более высоким приоритетом или при отклике на соответствующий сигнал REQ#.
4. Если арбитр выставляет сигнал GNT# агента и шина находится в состоянии "свободно", то агент должен включить выходные буферы линий AD[31::00], C/BE[3::0]# в пределах восьми тактов шины PCI и на один такт позже сигнала PAR. Рекомендуется два-три такта.

Временные задержки

1. Все цели обязаны выполнять начальную фазу данных транзакции (чтения или записи) в пределах 16 тактов от установки сигнала FRAME#.
2. Цель должна выполнять следующую фазу данных в пределах восьми тактов от завершения предыдущей фазы данных.
3. Мастер обязан выставлять сигнал IRDY# в пределах восьми тактов для любой фазы данных (начальной и последующей).

Правила выбора устройства

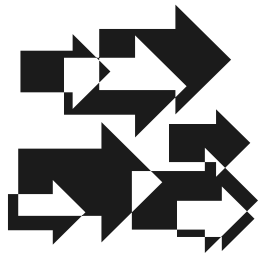
1. Цель должна выполнить полную дешифрацию перед управлением/установкой сигнала DEVSEL# или любого другого сигнала отклика цели.
2. Цель должна выставить сигнал DEVSEL# (требовать транзакции) до установки любого другого сигнала отклика.
3. Во всех случаях, кроме "Target-Abort", после выставления сигнала DEVSEL# цель не должна его снимать, пока не снят FRAME# (IRDY# выставлен), и пока не выполнится последняя фазы данных.

4. PCI-устройство является целью конфигурационной транзакции типа 0 (чтения или записи), только если выставлен ее сигнал IDSEL, и линии AD[1::0] содержат значение "00" во время фазы адреса команды.

Реализация проверки на четность

1. Четность проверяется согласно следующим правилам:
   - паритет вычисляется одинаково для всех PCI-транзакций независимо от типа или формы;
   - сумма "1" на линиях AD[31::00], C/BE [3::0]# и PAR равняется четному числу;
   - сумма "1" на линиях AD[63::32], C/BE [7::4]#, и PAR64 равняется четному числу;
   - генерация четности является обязательной и должна быть выполнена всеми PCI-совместимыми устройствами.
2. Только мастер поврежденной передачи данных может сообщать об ошибках четности данных программному обеспечению, используя отличные от сообщения по линии PERR# механизмы, т. е. через прерывания или по линии SERR#. В некоторых случаях мастер передает эту функцию мосту PCI-to-PCI, обрабатывающему отложенную запись в память.

ГЛАВА 7



Дополнительные возможности расширения

Расширение шины до 64 разрядов

Стандарт PCI поддерживает расширение шины до 64 разрядов. Реализация дополнительных 32 разрядов требует дополнительных 39 линий: REQ64#, ACK64#, AD[63::32], C/BE[7::4]# и PAR64. При обмене 32-разрядных агентов с 64-разрядными взаимодействие осуществляется без изменений. Устройства (агенты) по умолчанию должны функционировать в 32-разрядном режиме, если специально не установлен режим 64-битной транзакции. Основная идея заключается в обеспечении прозрачности 64-разрядных транзакций по отношению к 32-разрядным устройствам. 64-битная адресация не требует 64-битных адресных линий.

64-разрядный режим обмена между мастером и целью устанавливается динамически для каждой транзакции. Для этого мастер выставляет сигнал REQ64#, цель подтверждает согласование транзакции, выставляя сигнал ACK64#. Разрядность транзакции не изменяется до ее завершения. Сигнал ACK64# не должен быть выставлен, если в этой же транзакции не выставлен сигнал REQ64#. На линиях REQ64# и ACK64# должна быть внешняя нагрузка, чтобы обеспечить согласование при смешивании 32- и 64-битных агентов. Требования протокола шины и синхронизации остаются неизменными для 64-разрядной транзакции. 64-разрядные транзакции применяются только для доступа в память. Линия REQ64# не должна использоваться в транзакциях с командами *Interrupt Acknowledge* и *Special Cycle*. Увеличение разрядности для транзакций ввода-вывода неоправданно добавляющейся при этом сложностью, и поэтому 64-разрядные транзакции используются только для обращения к памяти [9, 18].

Все транзакции, осуществляющие обращение к памяти, и остальные обмены на шине функционируют одинаково независимо от разрядности передавае-

мых данных. 64-разрядные агенты могут передать от одного до восьми байт в каждой фазе данных, при этом допустимы все комбинации сигнала разрешения обращения к байтам. Как и для 32-разрядного режима, комбинация линий разрешения обращения к байтам может измениться в каждой фазе данных. Мастер, вводящий 64-битную транзакцию данных, должен использовать для адреса обращения двойное слово (Qword или 8 байтов, а линия AD[2] должна содержать "0" во время фазы адреса). При запросе мастером 64-битной передачи данных для цели допускается три варианта ответных действий:

1. Выполнить транзакцию, используя 64-битный тракт данных (сигнал ACK64# установлен).
2. Выполнить транзакцию, используя 32-битный тракт данных (сигнал ACK64# снят).
3. Выполнить одну 32-битную передачу данных (сигнал ACK64# снят, а сигнал STOP# установлен).

Первый вариант описывает действия цели, когда она в состоянии выполнить 64-битную транзакцию и отвечает мастеру, выставляя сигнал ACK64#. Передача данных осуществляется по всей шине данных, в каждой фазе данных может быть передано до 8 байтов. Транзакция протекает так же, как и на 32-битной шине, за исключением большего количества данных, передаваемых в каждой фазе данных.

Во втором варианте цель не может выполнить 64-битную передачу данных по указанному адресу, т. к., например, адресуемая область находится в различных пространствах. В этом случае мастер должен выполнить транзакцию в качестве 32-битного мастера, а не 64-битного. Мастер может начать транзакцию записи двумя способами, если на установку сигнала REQ64# цель не отвечает установкой ACK64#. Первым способом мастер освобождает старшие адресные линии и работает только с младшими 32 адресными линиями. Второй способ — мастер выдает все 64 бита данных на каждом четном двойном слове. На нечетном двойном слове мастер управляет одинаковыми данными на старшей и младшей частях шины.

В третьем варианте ответа 32-разрядная цель не может поддержать блок большей разрядности для данной транзакции. В этом случае цель выставляет не сигнал ACK64#, а сигнал STOP# и завершает транзакцию. Если же при этом снят сигнал TRDY# (тип завершения "Retry"), то мастер повторяет этот 64-битный запрос позже. Если сигнал TRDY# выставлен (тип завершения "Disconnect"), то мастер должен повторить запрос как 32-битный мастер, т. к. стартовый адрес находится теперь на нечетной границе. Если цель выполнила передачу данных так, что следующий стартовый адрес находится на четной границе двойного слова, то мастер имеет право запросить 64-битную передачу данных.

Правила генерирования четности для старших 32 битов 64-битной шины не изменяются по сравнению с правилами для младшей 32-битной шины. Сигнал PAR64 охватывает линии AD[63::32] и C/BE[7::4]# и имеет такую же синхронизацию и функциональное назначение, как сигнал PAR. Сигнал PAR64 должен быть действителен один такт после каждой фазы адреса на любой транзакции, в которой выставлен сигнал REQ64# и требуется для 64-битных фаз данных. На рис. 7.1 и 7.2 64-битный мастер запрашивает 64-битную транзакцию, используя одну фазу адреса. На рис. 7.1 представлена транзакция чтения, где цель отвечает установкой сигнала ACK64#, и данные передаются в 64-битных фазах данных. На рис. 7.2 представлена транзакция записи, где цель не отвечает установкой сигнала ACK64#, и данные переда-

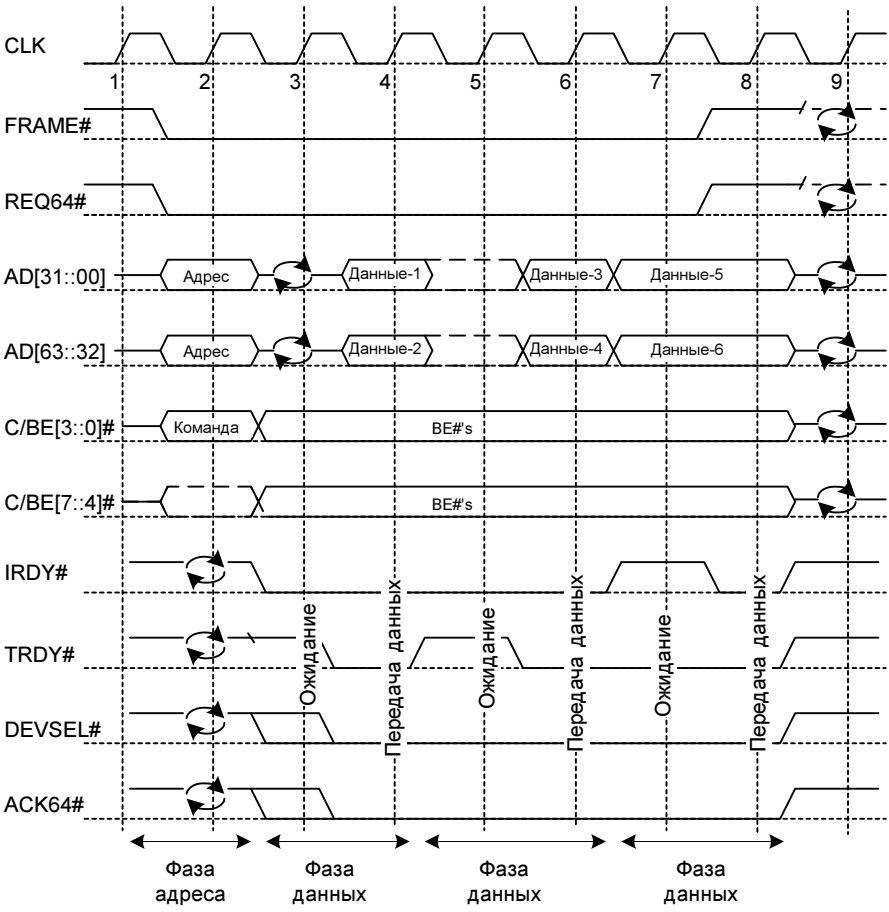


Рис. 7.1. 64-битный запрос чтения с 64-битной передачей

ются в 32-битных фазах данных (транзакция по умолчанию находится в 32-битном режиме).

Линии AD[63::32] и C/BE[7::4]# зарезервированы в течение фазы адреса одной транзакции. Линии AD[63::32] содержат данные, а линии C/BE[7::4]# содержат разрешение обращения к байтам для старших четырех байтов во время 64-битных фаз данных этих транзакций. Линии AD[63::32] и C/BE[7::4]# определены во время двух фаз адреса двойного цикла адреса (DAC) и во время 64-битных фаз данных. На рис. 7.1 показан запрос мастером 64-битной транзакции чтения установкой сигнала REQ64#. Цель принимает запрос, выставляя сигнал ACK64#. Продолжительность фаз данных увеличена обоими агентами. Оборотные циклы для 64-битных линий идентичны оборотным циклам для 32-битных линий.

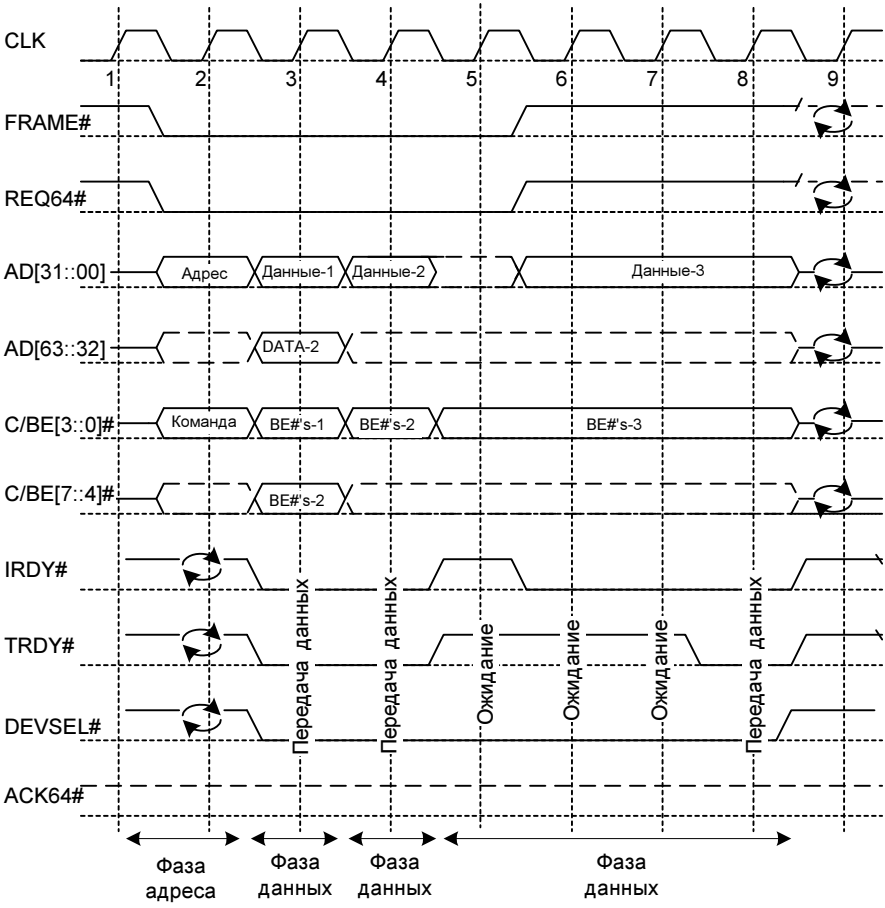


Рис. 7.2. 64-битный запрос записи с 32-битной передачей

На рис. 7.2 показан запрос мастером 64-битной транзакции. Цель доступа 32-битная и не соединена с линиями REQ64# или ACK64#. Сигнал ACK64# остается в снятом состоянии. Мастер преобразовывает транзакцию из 64- в 32-битную, следовательно, там может быть выставлен любой сигнал разрешения обращения к байтам для любой фазы данных транзакции. Поэтому все 32-битные цели должны быть в состоянии обрабатывать фазы данных без этого сигнала. Цель не должна использовать завершение с типом "Disconnect" или "Retry" из-за отсутствия сигнала разрешения обращения к байтам, но должна выставить сигнал TRDY# и выполнить фазу данных. При повторных транзакциях последующие фазы данных появляются как 32-битная передача.

64-битные передачи с одной фазой данных не могут быть эффективными. Мастер не обладает информацией о подтверждении 64-битной транзакции до установки сигналов ACK64# и DEVSEL#, следовательно, неизвестен такт, на котором снимается сигнал FRAME# для одной транзакции 64-битной фазы данных. Сигнал IRDY# должен оставаться в неактивном состоянии до разрешения сигнализации FRAME#. Одна 64-битная фаза данных, вероятно, будет разделена на две 32-битных, если цель 32-битная. Это означает, что две 32-битные фазы данных будут переданы с такой же или большей скоростью как одна 64-битная фаза.

Определение размерности шины

Определение размерности шины осуществляется во время инициализации системы. Сигнал REQ64# используется во время системного сброса для разделения участков, связанных с 64-битной шиной данных, от обычных. Линия REQ64# слотов плат расширения PCI, которые поддерживают только 32-битную шину данных, не должны соединяться с любыми другими слотами или устройствами (выводы REQ64# и ACK64# расположены в 32-битной части разъема). Каждый 32-битный разъем должен иметь индивидуальный нагрузочный резистор для линии REQ64# на системной плате. Линия ACK64# является шинной для всех 64-битных устройств и слотов на системной плате и подтягивается одним резистором, расположенным на системной плате. Сигнал линии ACK64# для каждого 32-битного слота должен быть снят либо путем соединения его с сигналом линии ACK64# для 64-битных устройств, либо индивидуальными нагрузочными резисторами на системной плате.

Линия REQ64# является шинной для всех устройств на системной плате (включая разъемы PCI-слотов), которые поддерживают 64-битную шину данных. Эта линия имеет один нагрузочный резистор на системной плате. Центральный ресурс должен переводить сигнал REQ64# в активное состояние во время установки сигнала RST#. Устройства, которые обнаруживают установленный сигнал REQ64# на переднем фронте сигнала RST#, соединены с 64-битной шиной данных и соответственно устройства, которые не обнару-

живают установку сигнала REQ64#, не соединены. Эта информация может использоваться компонентом для стабилизации плавающих входов во время функционирования. Для сигнала REQ64# имеются требования времени установления и удержания относительно неактивного состояния сигнала RST#. Пока выставлен сигнал RST#, сигнал REQ64# является асинхронным относительно тактовых сигналов CLK. Реализация 64-битной шины данных требует наличия на линиях AD[63::32], C/BE[7::4]# и PAR64 нагрузочных резисторов. Это объясняется тем, что они не используются в транзакциях с 32-битными устройствами и поэтому могут в плавающем состоянии достигать порогового уровня, вызывающего колебание или утечку питания через входной буфер. Подтягивающее напряжение должно обеспечиваться центральным ресурсом системной платы, а не платой расширения, чтобы гарантировать последовательное решение и избежать перегрузки из-за подтягивающего тока.

Когда 64-битная шина данных присутствует в устройстве, но не соединена (например, 64-битная плата расширения, вставленная в 32-битный PCI-слот), то PCI-устройство не должно содержать "дребезжащих" входов, не должно быть утечки питания через входной буфер. Данное требование может быть выполнено различными путями; например, смещая входной буфер или непрерывно управляя выходами, т. к. они ни с чем не связаны. Внешние резисторы на плате расширения или любые другие решения, нарушающие спецификацию входного тока утечки, запрещены. Пока сигнал RST# находится в активном состоянии, PCI-устройство переводит в плавающее состояние его выходные буферы для расширенной шины данных: AD[63::32], C/BE[7::4]# и PAR64. Если входные буферы устройства не должны находиться в плавающем состоянии, то компонент обязан управлять его входами все время, пока сигнал RST# находится в активном состоянии. После того как устройство обнаруживает снятие сигнала REQ64# на переднем фронте сигнала RST#, оно должно продолжить управление расширенной шиной для обеспечения защиты входных буферов устройства [9].

64-битная адресация

Шина PCI поддерживает адресацию памяти за пределами адресного пространства 4 Гбайт путем определения механизма для передачи 64-битного адреса от мастера транзакции к цели. Для 32- или 64-битного устройства не требуются дополнительные выводы, чтобы поддерживать 64-битную адресацию. Устройства, которые поддерживают только 32-битную адресацию, отображаются в нижнее 4-гигабайтное адресное пространство и работают прозрачно с устройствами, которые генерируют 64-битные адреса. Только транзакции памяти поддерживают 64-битную адресацию. Размерность шины адреса не зависит от размерности шины мастера или цели. Если и мастер и цель поддерживают 64-битную шину, то в одном такте теоретически мог бы

быть обеспечен полный 64-битный адрес. Но во всех случаях мастер должен использовать два такта для сообщения 64-битного адреса, т. к. размерность шины цели неизвестна во время фазы адреса. Стандарт транзакций шины PCI поддерживает 32-битный адрес, одиночный адресный цикл (Single Address Cycle — SAC), где адрес действителен для одного такта. Для поддержки передачи 64-битного адреса используется команда шины "двойной адресный цикл" (DAC). Эта команда сопровождается одной из допустимых команд шины, чтобы указать желаемую активность фазы данных для транзакции. Для DAC используются два такта, чтобы передать полный 64-битный адрес по линиям AD[31::00]. При использовании 64-битным мастером DAC (64-битная адресация) он должен обеспечить старшие 32 бита адреса на линиях AD[63::32] и соответствующую команду для транзакции на линиях C/BE [7::4]# во время обеих фаз адреса транзакции. Это предоставляет 64-битным целям дополнительное время для дешифрации транзакции.

На рис. 7.3 представлен двойной адресный цикл для транзакции чтения. В основной транзакции чтения SAC оборотный цикл следует за фазой адреса. В транзакции чтения DAC между стандартной фазой адреса и оборотным циклом вставлена дополнительная фаза адреса. На рис. 7.3 первая и вторая фаза адреса выполняется на тактах 2 и 3 соответственно. Оборотный цикл между фазой адреса и фазой данных отложен до такта 4.

Сигнал FRAME# должен быть выставлен на обеих фазах адреса (даже одиночных транзакций) с одиночными фазами данных. С целью выполнения отношений FRAME# — IRDY#, сигнал FRAME# не может быть снят, пока не выставлен IRDY#. Сигнал IRDY# не может быть выставлен, пока мастер не обеспечит данные на транзакции записи или пока он не будет готов принять данные на транзакции чтения. Дешифрация DAC потенциальной целью выполняется в момент, когда на линиях C/BE[3::0]# присутствует значение "1101" во время первой фазы адреса. Если 32-битная цель поддерживает 64-битную адресацию, то она сохраняет адрес, переданный по линиям AD[31::00], и готовится зафиксировать остальную часть адреса на следующем такте. Фактическая команда транзакции передана во время второй фазы адреса по линиям C/BE[3::0]#. 64-битной цели разрешено зафиксировать полный адрес на первой фазе адреса. Как только передан полный адрес и команда зафиксирована, цель определяет, является ли она целью транзакции и должен ли быть выставлен сигнал DEVSEL#. Агент, выполняющий вычитающую дешифрацию, обязан правильно обрабатывать задержку выбора устройства либо путем игнорирования всей транзакции, либо путем задержки установки его собственного сигнала DEVSEL#. Если мост поддерживает 64-битную адресацию, то он задержит установку его сигнала DEVSEL#.

Мастер двойного адресного цикла также задержит завершение транзакции с типом "Master-Abort" на один дополнительный такт. Выполнение моно-

польного доступа осуществляется одинаково для DAC и SAC. В любом случае сигнал LOCK# находится в неактивном состоянии во время фазы адреса (первый такт) и в активном во время второго такта (который является первой фазой данных для SAC и второй фазы адреса для DAC). Агенты, отслеживающие транзакции, обнаруживают, что заблокированный ресурс занят и цель "знает", что мастер запрашивает заблокированную операцию. Для цели, которая поддерживает и SAC и DAC, логика, которая обрабатывает сигнал LOCK#, является одинаковой.

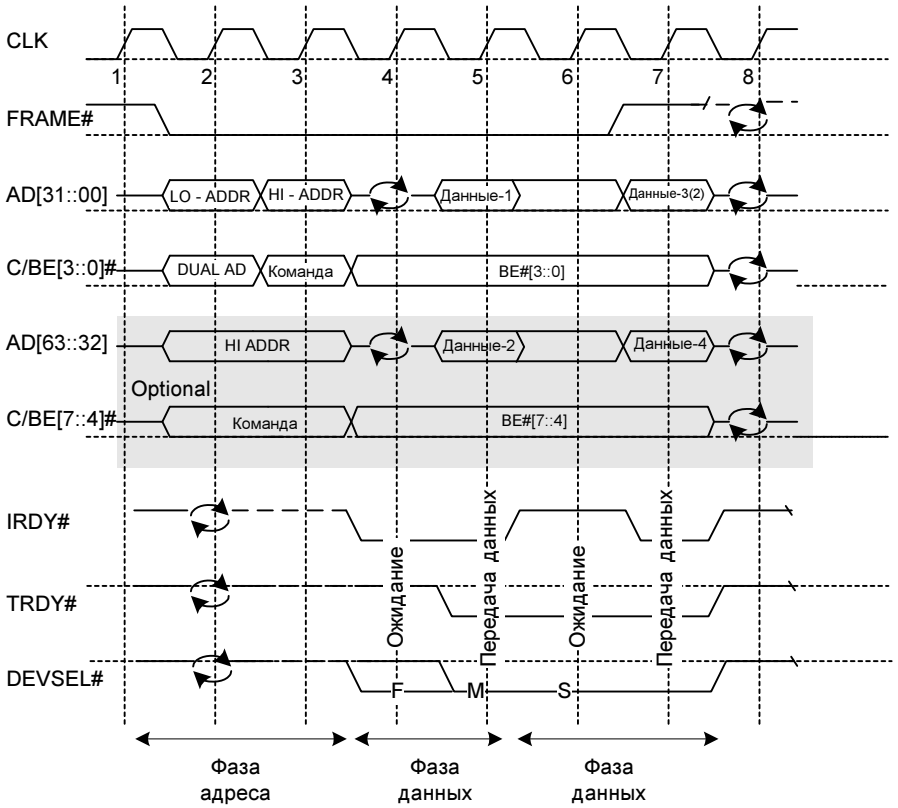


Рис. 7.3. 64-битный цикл двойного чтения адреса

Мастер передает 64-битный адрес, как это показано на рис. 7.3, независимо от того, поддерживает цель 64-битную шину или нет. Выделенная серым цветом часть временной диаграммы, показанная на рис. 7.3, используется только для случая поддержки мастером 64-битной шины. Мастер управляет всеми адресными линиями и обеими командами (DAC соответствует "1101" на линиях C/BE[3::0]# и фактическая команда шины на C/BE[7::4]#) во время на-

чальной фазы адреса. На второй фазе адреса мастер управляет старшим адресом на линиях AD[31::00] (и AD[63::32]), в то время как команда шины управляется на линиях C/BE[3::0]# (и C/BE[7::4]#). Мастер не может определить, поддерживает ли цель 64-битную шину данных, пока не был передан полный адрес, и поэтому должен по умолчанию предполагать 32-битную цель. Если и мастер и цель поддерживают 64-битную шину, то 64-битная адресация не вызывает дополнительной задержки при определении сигнала DEVSEL#, т. к. вся требуемая информация для дешифрации команды находится в первой фазе адреса. Например, 64-битная цель, обычно выполняющая среднюю дешифрацию DEVSEL# для SAC, может дешифрировать полный 64-битный адрес от 64-битного мастера во время первой фазы адреса DAC и выполнить быструю дешифрацию DEVSEL#. Если мастер или цель не поддерживают 64-битную шину данных, то будет введен один дополнительный такт задержки. Мастер, который поддерживает 64-битную адресацию, должен выполнить SAC вместо DAC, когда старшие 32 бита адреса содержат ноль. Это позволяет мастерам, генерирующим 64-битные адреса, связываться с 32-битными целями посредством SAC. Тип адресации (SAC или DAC) зависит от того, находится ли адрес в нижнем 4-гигабайтном диапазоне адреса или нет, а вовсе не от размерности шины цели. Адресуемая 64-битная цель должна действовать подобно адресуемой 32-битной цели (ответить на транзакции SAC) при отображении в нижнее 4-гигабайтное адресное пространство. Если 32-битный мастер осуществляет доступ к целям, отображенным выше этого 4-гигабайтного адресного пространства, то мастер должен поддерживать 64-битную адресацию, используя DAC [9, 18].

Работа шины на частоте 66 МГц

Введение

Данный раздел базируется на спецификации для частоты 33 МГц, и здесь описаны только требования, правила и параметры, отличные от стандартной спецификации. Шина PCI 66 МГц представляет собой совместимое надмножество PCI для функционирования на частоте 66 МГц. Для работы на этой частоте мосты и периферийные устройства должны обладать малым временем задержки и большой полосой пропускания. Допускается совместное использование с шиной PCI 33 МГц, для работы с низкоскоростной периферией.

Различия между шинами 33 и 66 МГц минимальны. Обе шины используют одинаковые протоколы, определения сигналов и расположение разъема. С целью идентификации PCI-устройства для частоты 66 МГц добавляется один статический сигнал путем переопределения существующего вывода земли. В регистр пространства конфигурации STATUS добавляется один бит.

Характеристики постоянного тока, пределы точки управления шинных драйверов соответствуют характеристикам шинных драйверов для шины PCI 33 МГц; но из-за увеличения частоты предъявляются другие требования к параметрам синхронизации и изменяется переопределение условий измерения. В результате шины на 66 МГц могут поддерживать меньшую нагрузку и длину линии. Устройства, предназначенные для работы на частоте 66 МГц при связывании их с шиной на 33 МГц, работают на частоте 33 МГц. Аналогично шина частотой 66 МГц при соединении с устройствами, предназначенными для работы на частоте 33 МГц, работает на частоте 33 МГц. Программные модели шин частотой 66 и 33 МГц идентичны, заголовки пространства конфигурации и коды классов идентичны. Агенты и мосты содержат бит статуса 66 МГц PCI, т. е. признак совместимости с шиной на 66 МГц.

Таким образом, в данном разделе определены аспекты шины PCI на 66 МГц, которые отличаются от стандартных параметров шины, включая информацию относительно устройства и поддержку мостов. Идентичные параметры, описанные ранее, не будут повторены [9, 18].

Пространство конфигурации

Совместимость PCI-устройства с шиной на 66 МГц идентифицируется с помощью флага 66MHZ_CAPABLE, флаг доступен только для чтения и расположен в бите 5 регистра STATUS пространства конфигурации. Установка данного бита означает, что устройство способно к работе на 66 МГц.

Архитектура агента

Все агенты, совместимые с шиной на 66 МГц, должны поддерживать флаг 66MHZ_CAPABLE, расположенный в бите 5 регистра STATUS пространства конфигурации этого агента. Если бит **66MHZ_CAPABLE** установлен (табл. 7.1), то он указывает, что агент может работать в режиме 66 МГц PCI.

Протокол шины

Определение вывода 66MHZ_ENABLE (M66EN)

Вывод 49B на PCI-разъеме обозначен как "M66EN". Платы расширения и сегменты шины, не способные к работе на частоте 66 МГц, должны соединить этот вывод с землей. Сегмент системной платы, предназначенный для работы на частоте 66 МГц, должен обеспечить один нагрузочный резистор на выводе M66EN. Линия M66EN шинно соединена со всеми разъемами и компонентами, предназначенными для использования только на системной плате, которые содержат вывод M66EN. Цепь синхронизации 66 МГц должна

соединяться с выводом M66EN, чтобы генерировать такты соответствующей частоты для сегмента (от 33 до 66 МГц, если выставлен сигнал M66EN, и от 0 до 33 МГц, если сигнал M66EN снят).

Если агенту, совместимому с шиной на 66 МГц, требуется информация о частоте синхронизации (например, для шунтирования ФАПЧ), то допускается использовать вывод M66EN как вход. Если агент может функционировать без информации о частоте синхронизации, то вывод M66EN может быть отсоединен.

Таблица 7.1. *Комбинации шин и агентов*

Бит шины 66MHZ_CAPABLE[1]	Бит агента 66MHZ_CAPABLE	Описание
0	0	Агент, работающий на частоте 33 МГц, размещен на шине с частотой 33 МГц
0	1	Агент, работающий на частоте 66 МГц, размещен на шине с частотой 33 МГц[2]
1	0	Агент, работающий на частоте 33 МГц, размещен на шине с частотой 66 МГц
1	1	Агент, работающий на частоте 66 МГц, размещен на шине с частотой 66 МГц

Задержки

Шина PCI частотой 66 МГц предназначена для устройств с малым временем задержки, начальное время задержки цели не должно превышать 16 тактов.

Электрическая спецификация

Различия электрической спецификации не сильно отличаются от аналогичных для шины с частотой 33 МГц. Определены электрические характеристики и ограничения компонентов, системных плат и плат расширения, предназначенных для работы на частоте 66 МГц, включая назначения выводов разъема. Все требования и характеристики для шины 33 МГц выполняются и для шины 66 МГц, за исключением специально оговоренных.

---

[1] Бит статуса шины **66MHZ_CAPABLE** размещен в регистре STATUS пространства конфигурации моста.

[2] Это состояние может привести к генерации конфигурационным ПО предупреждения о том, что плата расширения установлена в неправильный слот и должна быть удалена в корзину.

Наиболее существенными отличиями являются:

- ❒ шина с частотой 66 МГц функционирует только в среде 3,3 В;
- ❒ параметры синхронизации были рассчитаны для частоты 66 МГц;
- ❒ условия нагрузочных испытаний переменного тока были изменены.

Направления перехода к частоте 66 МГц

Шина с частотой 66 МГц использует стандартный протокол шины PCI; по сравнению со стандартной шиной повышена максимальная частота тактов. В одном сегменте шины могут сосуществовать устройства, разработанные для частот 33 и 66 МГц. В этом случае сегмент шины будет работать на частоте 33 МГц. Для обеспечения обратной совместимости устройства частотой 66 МГц имеют такие же спецификации постоянного тока и пределы переменного тока, как и устройства частотой 33 МГц. Но для частоты 66 МГц требуется изменение параметров синхронизации, как показано на рис. 7.4.

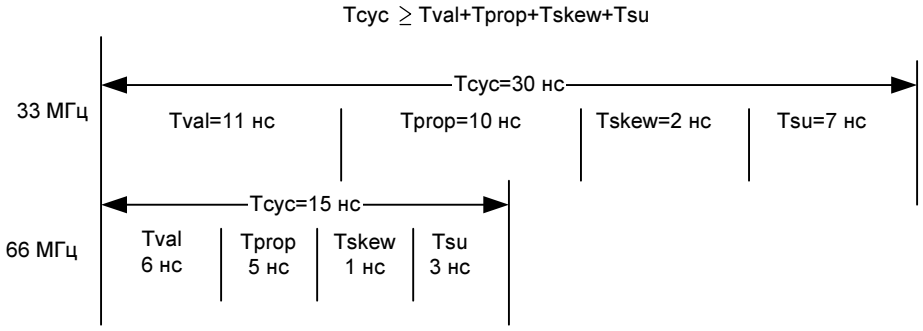


Рис. 7.4. Анализ параметров синхронизации для частот 33 и 66 МГц

Поскольку требования переменного тока не изменились для частоты 66 МГц, то устройства, разработанные для частоты 66 МГц, должны функционировать на шине с частотой 33 МГц. Поэтому устройства с рабочей частотой 66 МГц должны выполнить, насколько это возможно, требования для обеих частот.

Сигнальная среда

Сегмент системной платы, предназначенный для функционирования на частоте 66 МГц, должен использовать разъем с напряжением 3,3 В с дополнительным механическим ключом. Поэтому сегменты системной платы частоты 66 МГц принимают платы расширения 3,3 В или универсальные платы расширения, а платы расширения с питанием 5 В не поддерживаются. Параметры драйверов шины с частотой 33 МГц определяются их вольт-амперными

характеристиками (ВАХ), выходные буферы для шины 66 МГц определяются в терминах их точек управления переменного тока, постоянного тока, параметров синхронизации и скоростей нарастания. Минимальное значение точки управления переменного тока определяет приемлемое пороговое напряжение и должно быть достигнуто в пределах максимального значения времени T_{val}. Максимальное значение точки управления переменного тока ограничивает диапазон управляющего напряжения в системе. Точка управления постоянного тока определяет условия устойчивого состояния. Минимальное значение скорости нарастания и параметры синхронизации гарантируют функционирование на частоте 66 МГц. Максимальное значение скорости нарастания минимизирует шум системы. Требования данных параметров в такой совокупности обеспечивает более краткое определение для выходного буфера.

Спецификация постоянного тока

Эта спецификация идентична спецификации постоянного тока для PCI-компонентов.

Спецификация переменного тока

Параметры переменного тока среды для шины с частотой 66 МГц для напряжения 3,3 В приведены в табл. 7.2. Определения вольт-амперных характеристик, скорости нарастания, тока переключения высокого (или низкого) уровня, тока замыкания высокого (низкого) уровня приведены в *главе 8* [9, 21].

Таблица 7.2. *Спецификация переменного тока*

Символ	Параметр	Условия	Мин.	Макс.	Ед. изм.	Прим.
Точки управления переменного тока						
$I_{oh(AC,\ min)}$	Ток переключения высокого уровня, минимум	$V_{out} = 0{,}3V_{cc}$	$-12V_{cc}$	—	мА	1
$I_{oh(AC,\ max)}$	Ток переключения высокого уровня, максимум	$V_{out} = 0{,}7V_{cc}$	—	$-32V_{cc}$	мА	
$I_{ol(AC,\ min)}$	Ток переключения низкого уровня, минимум	$V_{out} = 0{,}6V_{cc}$	$16V_{cc}$		мА	1
$I_{ol(AC,\ max)}$	Ток переключения низкого уровня, максимум	$V_{out} = 0{,}18V_{cc}$	—	$38V_{cc}$	мА	

Таблица 7.2 *(окончание)*

Символ	Параметр	Условия	Мин.	Макс.	Ед. изм.	Прим.
Точки управления постоянного тока						
V_{OH}	Выходное напряжение высокого уровня	I_{out} = –0,5 мА	0.9 V_{cc}	—	В	2
V_{OL}	Выходное напряжение низкого логического уровня	I_{out} = 1,5 мА	—	0.1 V_{cc}	В	2
Скорость нарастания						
t_r	Выходная скорость нарастания	$0.3V_{cc}$ – $0.6V_{cc}$	1	4	В/нс	3
t_f	Выходная скорость понижения	$0.6V_{cc}$ – $0.3V_{cc}$	1	4	В/нс	3
Ток замыкания						
I_{ch}	Ток замыкания высокого уровня	V_{cc} + 4 > V_{in} ≥ V_{cc} + 1	25 + (V_{in} – V_{cc} – 1)/0.015	—	мА	
I_{cl}	Ток замыкания низкого уровня	–3 < V_{in} ≤ –1	–25 + (V_{in} + 1)/0.015	—	мА	

Примечания

1. Допускается применение характеристик токов переключения для линий REQ# и GNT#, равных половине от определенного здесь значения; т. е. для этих линий могут использоваться выходные драйверы, настроенные на половину от значения данного уровня. Эти требования не применяются к линиям CLK и RST#, которые являются системными выводами. Параметр "ток переключения высокого уровня" не относится к выводам SERR#, PME#, INTA#, INTB#, INTC# и INTD#, которые являются выходами с открытыми коллекторами.
2. Значения постоянного тока определены в табл. 8.3 и включены здесь для полноты.
3. Этот параметр должен интерпретироваться как скорость накопления уровня через определенный диапазон, а не как мгновенная скорость в любой точке внутри диапазона. Указанная нагрузка (см. рис. 7.10) является необязательной, разработчик может реализовать этот параметр с ненагруженным выходом. Тем не менее, несмотря на рекомендательный характер, требуется соблюдение максимальных и минимальных параметров. Скорость нарастания не применяется к выходам с открытым коллектором [9].

Максимальные значения переменного тока и защита устройств

Максимальные значения параметров переменного тока и защита устройств идентична спецификации постоянного тока для PCI-компонентов.

Временные параметры

Спецификация тактовых импульсов

Тактовый сигнал заданной формы, или отвечающий заданным параметрам, должен быть подан на соответствующий вход каждого компонента, предназначенного для функционирования на частоте 66 МГц. Применительно к платам расширения, соответствие со спецификацией синхронизации относится к компоненту платы расширения, а не к слоту. На рис. 7.5 приведена форма тактового сигнала, и требуемые точки измерения для среды передачи сигналов 3,3 В. Табл. 7.3 суммирует параметры тактового сигнала.

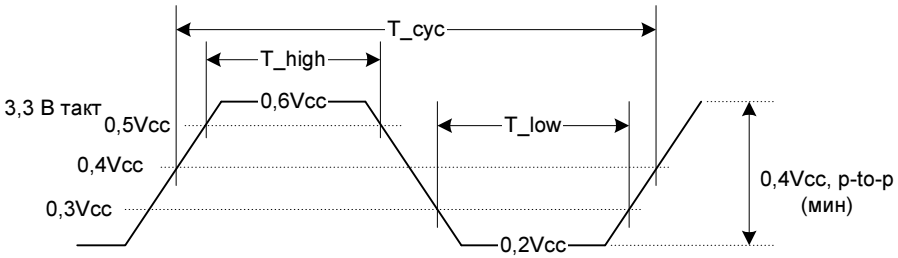


Рис. 7.5. Форма тактового сигнала среды 3,3 В

Расширение спектра разрешено в пределах, указанных в табл. 7.3.

Таблица 7.3. *Спецификация синхроимпульса*

Символ	Параметр	66 МГц		33 МГц[4]		Ед. изм.	Прим.
		Мин.	Макс.	Мин.	Макс.		
T_{cyc}	Период сигнала CLK	15	30	30	∞	нс	1, 3
T_{high}	Время нахождения сигнала CLK на высоком уровне	6		11		нс	
T_{low}	Время нахождения сигнала CLK на низком уровне	6		11		нс	
—	Время нарастания сигнала CLK	1.5	4	1	4	В/нс	2

Таблица 7.3 *(окончание)*

Символ	Параметр	66 МГц		33 МГц[4]		Ед. изм.	Прим.
		Мин.	Макс.	Мин.	Макс.		
Требования расширения спектра							
f_{mod}	Частота модуляции	30	33	—	—	кГц	
f_{spread}	Расширения частоты	–1	0			%	

Примечания

1. Все PCI-компоненты 66 МГц должны работать с любой частотой тактов вплоть до 66 МГц. Требования сигнала CLK изменяются в зависимости от того, превышает ли тактовая частота 33 МГц.

   Эксплуатационные параметры устройства на частотах 33 МГц или меньше соответствуют спецификациям, описанным в *главе 8*. Частота тактов может быть изменена в любое время в течение функционирования системы, пока уровни тактов остаются "чистыми" (монотонными) и не превышают пределы минимального периода такта и нахождения на верхнем и нижнем уровне. Такт может быть остановлен только на низком логическом уровне.

   В диапазоне тактовых частот от 33 и до 66 МГц частота не может изменяться, за исключением установки сигнала RST# или использования методов расширения спектра.

2. Время нарастания определено в терминах скорости нарастания уровня, измеряемое в В/нс. Эта скорость нарастания измеряется для всей амплитуды тактового сигнала, как показано на рис. 7.5. Скорость нарастания синхроимпульса измеряется в цепи, показанной на рис. 7.10.

3. Не допускается выход за минимальное значение периода даже для отдельно взятого такта; т. е. для всех синхроимпульсов системы.

4. Эти значения дублируют значения, рассмотренные в *главе 8*, и включены здесь для сравнения.

Расширение спектра тактовых сигналов (SSC[3])

Изменение частоты разрешается только в меньшую сторону от максимального значения тактовой частоты (т. к. не допускается выход за пределы минимального значения периода тактов синхросигнала, приведенного в табл. 7.3). Если используется фазовая автоподстройка частоты (ФАПЧ) для отслеживания тактовых импульсов, она должна успевать реагировать на SSC-модуляцию, чтобы не накопить чрезмерную разницу фаз между входом ФАПЧ и выходным тактом (обычно называемую как "перекос отслеживания SSC"). Размер перекоса зависит от полосы пропускания ФАПЧ, угла фазы на

[3] SSC — Spread Spectrum Clocking.

частотах 30—33 кГц и объема расширения. Желательно максимизировать полосу пропускания и/или уменьшить угол фазы, чтобы минимизировать перекос отслеживания; в противном случае должно быть минимизировано изменение частоты [9].

Временные параметры

Временные параметры для частоты 66 и 33 МГц представлены в табл. 7.4.

Таблица 7.4. *Временные параметры 66 и 33 МГц*

Символ	Параметр	66 МГц		33 МГц[7]		Ед. изм.	Прим.
		Мин.	Макс.	Мин.	Макс.		
T_{val}	Задержка между CLK и шинными сигналами	2	6	2	11	нс	1, 3, 4, 8
$T_{val(ptp)}$	Задержка между CLK и двухточечными сигналами	2	6	2	12	нс	1, 3, 4, 8
T_{on}	Задержка перехода сигнала из плавающего состояния (Float) в активное (Active)	2		2		нс	1, 8, 9
T_{off}	Задержка перехода сигнала из активного состояния (Active) в плавающее (Float)		14		28	нс	1, 9
T_{su}	Время активного состояния сигнала на входе до момента установки сигнала CLK — для шинных сигналов	3		7		нс	2, 3, 10
$T_{su(ptp)}$	Время активного состояния сигнала на входе до момента установки сигнала CLK — для двухточечных сигналов	5		10, 12		нс	2, 3
T_h	Время удержания сигнала на входе в активном состоянии с момента установки сигнала CLK	0		0		нс	2
T_{rst}	Время активности сигнала RST# после подачи питания	1		1		мс	5

Таблица 7.4 *(окончание)*

Символ	Параметр	66 МГц		33 МГц[7]		Ед. изм.	Прим.
		Мин.	Макс.	Мин.	Макс.		
$T_{rst-clk}$	Время активности сигнала RST# после установления CLK	100		100		мкс	5
$T_{rst-off}$	Задержка перехода сигнала RST# из активного состояния в плавающее		40		40	нс	5, 6
t_{rrsu}	Время установления сигнала REQ64# после RST#	$10T_{cyc}$		10Tсус		нс	
t_{rrh}	Время удержания сигнала RST# после REQ64#	0	50	0	50	нс	
T_{rhfa}	Время между высоким состоянием сигнала RST# до первого конфигурационного доступа	225		225		такт	
T_{rhff}	Время между высоким состоянием сигнала RST# до первого выставления сигнала FRAME#	5		5		такт	

Примечания

1. См. условия измерения выбора времени на рис. 7.6. Все управляемые переходы сигнала управляются к уровням Voh или Vol в пределах времени Tсус.
2. См. условия временных измерений на рис. 7.7.
3. Сигналы REQ# и GNT# являются двухточечными. Их временные параметры отличаются от шинных сигналов. Сигналы GNT# и REQ# имеют время установления 5 нс на частоте 66 МГц. Все другие сигналы являются шинными.
4. Максимальные значения измерены с нагрузкой цепи, показанной на рис. 7.8 и 7.9. Минимальные значения временных параметров измерены относительно выводов компонента с нагрузкой цепи, показанной на рис. 7.10.
5. Если выставлен сигнал M66EN, то сигнал CLK является устойчивым при условии соответствия требованиям тактового сигнала. Сигнал RST# выставляется и снимается асинхронно относительно CLK.
6. Все выходные драйверы должны находиться в плавающем состоянии во время активности сигнала RST#.

7. Эти значения дублируют значения, приведенные в *главе 8*, и включены здесь для сравнения.
8. Когда выставлен сигнал M66EN, минимальное значение для величин Tval (минимум), Tval (ptp) (минимум) и Ton может быть уменьшено до 1 нс, если при этом обеспечивается минимальное значение в 2 нс при снятии сигнала M66EN.
9. С целью измерения времени перехода сигнала из плавающего состояния в активное и обратно, считается, что вывод находится в высокоимпедансном состоянии или отключен, когда полный ток, проходящий через вывод компонента, меньше или равен току утечки.
10. Время установления сигнала на входе применяется, когда устройство не управляет выводом. Устройства не могут управлять и принимать сигналы одновременно.

Условия испытаний и измерений

Условия испытаний и измерений представлены на рис. 7.6—7.10, а параметры условий измерения в табл. 7.5.

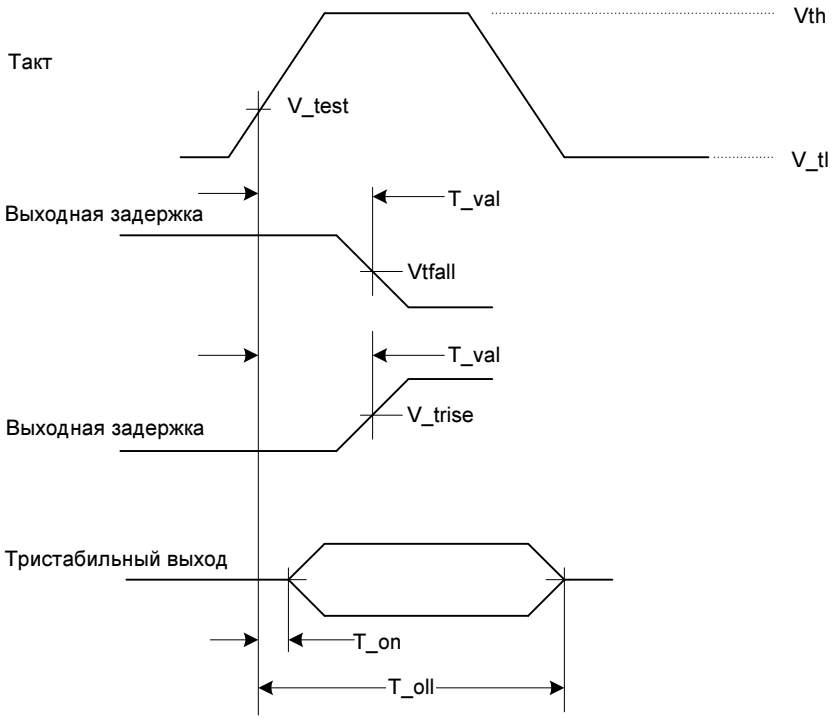


Рис. 7.6. Условия временных измерений выхода

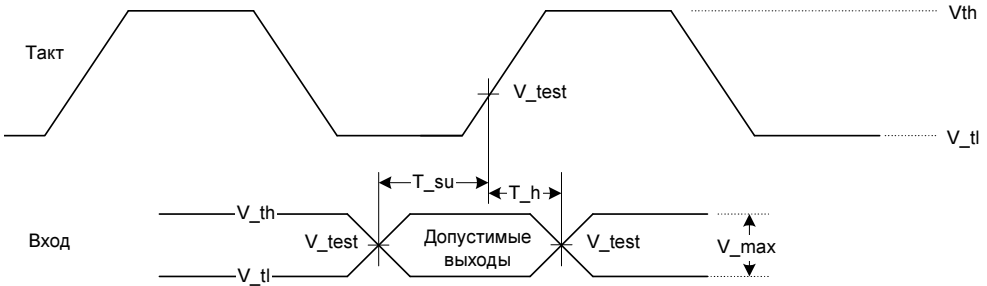


Рис. 7.7. Условия временных измерений входа

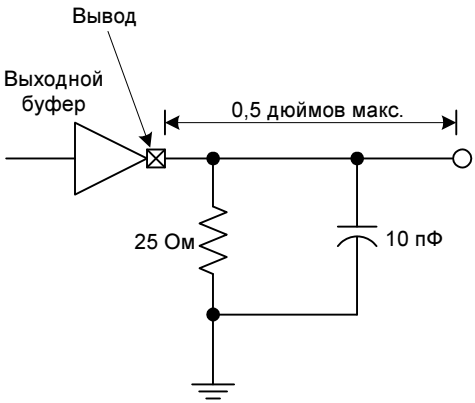


Рис. 7.8. Измерение $T_{val(max)}$ восходящего уровня

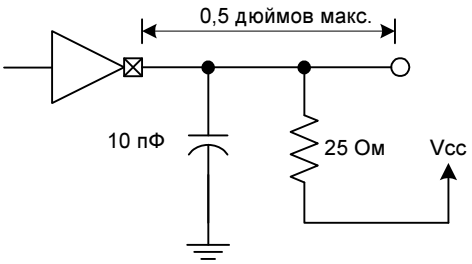


Рис. 7.9. Измерение $T_{val(max)}$ падающего уровня

Таблица 7.5. *Параметры условий измерения*

Символ	3,3 В среда	Ед. изм.	Прим.
V_{th}	$0,6V_{cc}$	В	1
V_{tl}	$0,2V_{cc}$	В	1
V_{test}	$0,4V_{cc}$	В	
V_{trise}	$0,285V_{cc}$	В	2
V_{tfall}	$0,615V_{cc}$	В	2
V_{max}	$0,4V_{cc}$	В	1
Скорость нарастания входного сигнала	1,5	В/нс	3

Примечания

1. Тестовое входное напряжение для среды передачи 3,3 В приведено с превышением на $0,1V_{cc}$. V_{max} определяет максимальную амплитуду сигнала, допустимую для измерения входного времени. Заводские испытания могут использовать различные значения напряжения, но результаты должны коррелировать с этими параметрами.
2. Величины Vtrise и Vtfall — рекомендованные напряжения только для временных измерений. Разработчики системы с шиной на 66 МГц должны разработать выходные буферы для линии 25 Ом таким образом, чтобы после первого отражения гарантировались правильные входные уровни.
3. Характеристика выходов и измерение на выводе компонента с нагрузкой показаны на рис. 7.10. Скорость изменения входного сигнала измеряется между значениями напряжений $0,2V_{cc}$ и $0,6V_{cc}$.

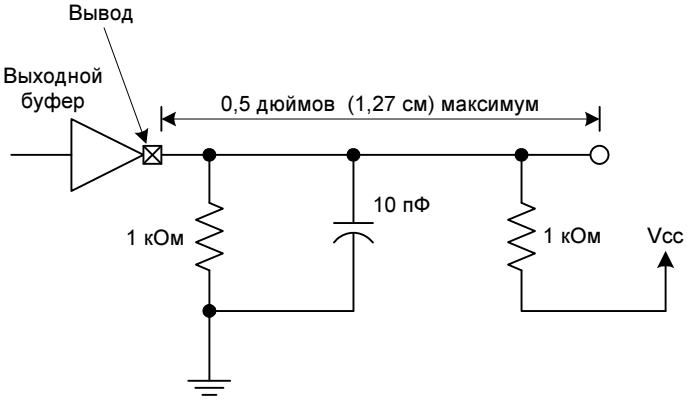


Рис. 7.10. Измерение $T_{val(min)}$ и скорости нарастания

Спецификация, обеспечиваемая производителем

Идентична спецификации, приведенной в *главе 8*.

Рекомендации по расположению выводов

Спецификации идентичны спецификации, приведенной в *главе 8*. Должны быть выполнены физические требования и электрическая спецификация для шины с частотой 66 МГц, но разработчику разрешено изменять предложенное расположение выводов.

Рекомендации синхронизации

На частоте 66 МГц рекомендуется выполнять тактирование следующим образом. Подключение тактового импульса 66 МГц как двухточечного сигнала от индивидуальных драйверов с малым перекосом синхронизации к системной плате и плате расширения сильно снизит эффекты отражения сигнала и оптимизирует целостность тактового сигнала. Кроме этого, будет уменьшен перекос синхроимпульса. На рис. 7.11 показан пример рекомендуемого соединения с линией тактовых сигналов различных устройств при функционировании на частоте 66 МГц [9].

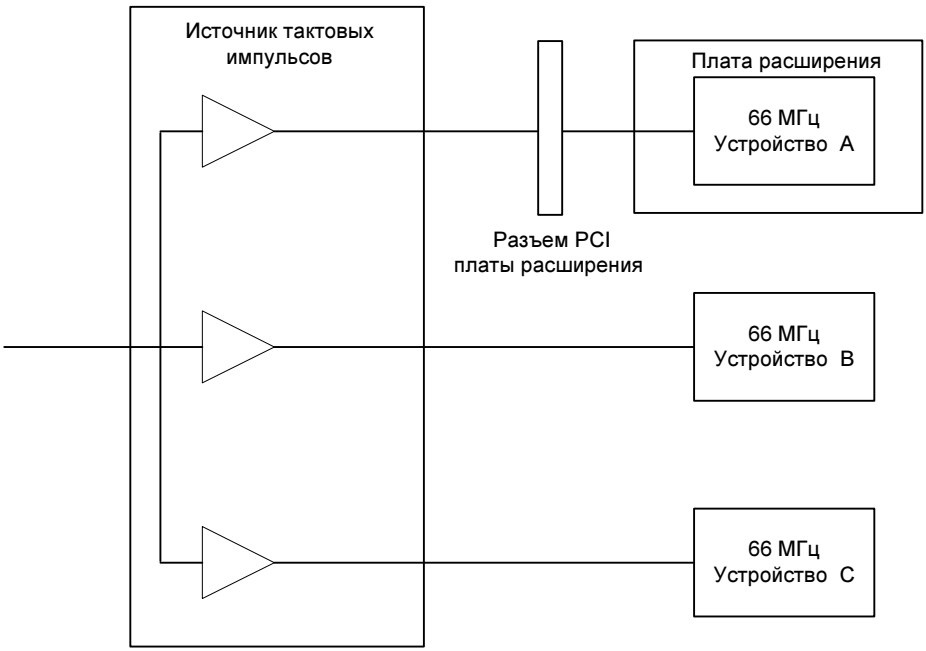


Рис. 7.11. Рекомендуемый метод тактирования

Спецификация системной платы

Перекос синхронизации

Максимально допустимый перекос синхронизации, включая "дрожание", составляет 1 нс. Данное значение применяется не только в одиночной пороговой точке, но и во всех точках на уровне такта в диапазоне переключения, определенном в табл. 7.6 и на рис. 7.12. Максимальный перекос измеряется между любыми двумя компонентами, а не между разъемами. Чтобы правильно оценить перекос такта, проектировщик системы должен принять во внимание распределение тактов на плате расширения.

Таблица 7.6. *Параметры перекоса синхронизации*

Символ	Среда 66 МГц 3,3 В	Среда 33 МГц 3,3 В	Ед. изм.
V_{test}	$0{,}4V_{cc}$	$0{,}4V_{cc}$	В
T_{skew}	1 (макс.)	2 (макс.)	нс

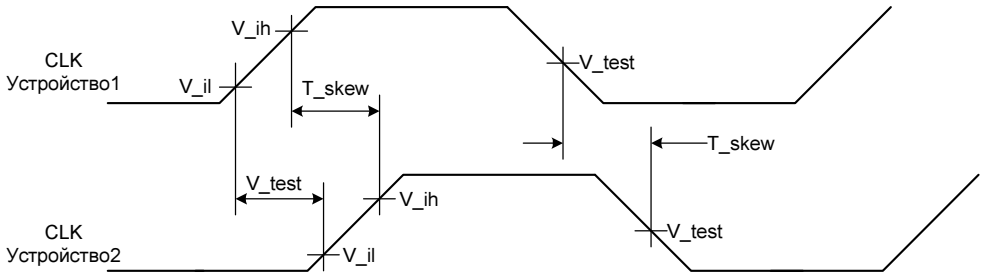


Рис. 7.12. Параметры перекоса тактовых импульсов

Системный сброс

Системный сброс идентичен требованиям, рассмотренным в *главе 9*.

Нагрузка

Для шины с частотой 66 МГц требуется отдельный нагрузочный резистор на системной плате для вывода M66EN.

Питание

Требования питания, последовательность активизации шин питания и развязка идентичны требованиям, рассмотренным в *главе 9*.

Распределение временных параметров

Полный период такта может быть разделен на четыре части. Допустимая выходная задержка (T_{val}) и входное время установки (T_{su}) определяется спецификацией компонента. Суммарный перекос синхронизации (T_{skew}) и максимальное время распространения по шине (T_{prop}) являются параметрами системной платы. T_{prop} является параметром системы, который может быть косвенно определен вычитанием временных параметров компонентов из периода такта. В табл. 7.7 приведено временное распределение для нескольких частот шины.

Таблица 7.7. *Временные параметры для нескольких частот*

Элемент	33 МГц	66 МГц	50 МГц[1]	Ед. изм.	Прим.
T_{cyc}	30	15	20	нс	
T_{val}	11	6	6	нс	
T_{prop}	10	5	10	нс	2
T_{su}	7	3	3	нс	
T_{skew}	2	1	1	нс	3

Примечания

1. Значения для частоты 50 МГц приведены только в качестве примера.
2. Данный параметр рассчитывается, остальные параметры установлены. Таким образом, замедление частоты шины позволяет разработчикам получить дополнительное расстояние или добавить дополнительные нагрузки. Спецификации компонентов обязаны гарантировать функционирование на частоте 66 МГц.
3. Здесь определен полный перекос, учитывающий все источники рассогласования. При использовании методов расширения спектра сигналов максимальный перекос такта на входе устройства платы расширения включает перекос отслеживания SSC.

Физические требования

Четырехслойные системные платы

Требования, предъявляемые к четырехслойным системным платам, идентичны требованиям, рассмотренным в *главе 9*.

Назначение выводов разъема

Единственным различием между разъемами, предназначенным для шин, работающих на частоте 33 и 66 МГц, является наличие вывода M66EN (вы-

вод 49B) для сегментов шины с частотой 66 МГц. Реализации, не способные к работе на частоте 66 МГц, должны соединить этот вывод с землей. Линия M66EN шинно соединяется со всеми разъемами в пределах одного логического сегмента шины при поддержке функционирования на частоте 66 МГц. Цепь "подтягивается" нагрузочными резисторами величиной 5 кОм до значения V_{cc}. Кроме того, эта цепь соединена с выводом M66EN входа компонентов, расположенных в том же самом логическом сегменте шины системной платы. Этот сигнал является статическим. Для обеспечения обратного контура переменного тока в пределах 0,25 дюймов от вывода M66EN должны быть расположены конденсаторы емкостью 0,01 мкФ. Эти конденсаторы включаются для каждого такого разъема платы расширения и должны развязать линию M66EN от земли. Любой присоединенный компонент или установленная плата расширения, которая не поддерживает частоту 66 МГц, должна "подтягивать" цепь M66EN к входному уровню Vil. Остальные компоненты, платы расширения, и логический сегмент шины с источником тактовых сигналов, таким образом, сигнализируют о работе на частоте 33 МГц [9].

Спецификация платы расширения

Единственным отличием плат расширения, функционирующих на частотах 33 и 66 МГц, является наличие вывода M66EN (вывод 49B). В устройствах, не способных к работе на частоте 66 МГц, этот вывод должен быть соединен с землей (GND). При реализации плат расширения, функционирующих на частоте 66 МГц, вывод M66EN должен быть развязан с землей через конденсаторы емкостью 0,01 мкФ. Конденсаторы должны располагаться в пределах 0,25 дюймов от края контакта, чтобы выполнить обратный контур переменного тока. Если вывод M66EN "подтянут" к уровню входа V_{il}, то это указывает, что плата расширения должна работать на частоте 33 МГц [9].

Поддержка SMBus

Интерфейс SMBus основан на спецификации System Management Bus Specification (спецификации SMBus 2.0). Этот двухпроводный последовательный интерфейс характеризуется низким питанием и малыми затратами на программное обеспечение, что делает его применимым для функций управления системой при малой полосе пропускания [9, 16].

Возможности, реализуемые интерфейсом SMBus:

- ❒ поддержка клиентских технологий управления;
- ❒ поддержка серверных технологий управления;
- ❒ поддержка тепловых датчиков и других контрольно-измерительных устройств на платах расширения;

- идентификация платы расширения, когда шина находится в состоянии B3 или когда PCI-устройство находится в состоянии D3hot или D3cold, как определено в спецификации "PCI Power Management Interface Specification".

Требования системы SMBus

Интерфейсы SMBus-устройства должны соответствовать электрическим требованиям и требованиям протокола, которые определены в спецификации SMBus 2.0. Соединения интерфейса SMBus в PCI-разъеме являются необязательными, но при поддержке интерфейса SMBus должны быть выполнены все описанные далее требования.

Питание

Рекомендуется подведение с системной платы вспомогательного питания 3,3 В к разъему PCI (вывод 14A). Это позволяет плате расширения поддерживать функциональные возможности SMBus, когда система находится в состоянии пониженного потребления питания.

Физический и логический сегмент SMBus

Физический сегмент SMBus определен как множество интерфейсов SMBus устройства, линии SMBCLK и SMBDAT, которые непосредственно связаны с друг другом (т. е. не связаны через повторитель SMBus или мост). Каждый физический сегмент шины должен соответствовать электрическим требованиям спецификации SMBus 2.0.

Логический сегмент SMBus определен как один или более физических сегментов SMBus, имеющих одно адресное пространство SMBus и одну арбитражную область SMBus. Несколько физических сегментов SMBus связываются в один логический сегмент SMBus посредством повторителей шины. Адресное пространство SMBus представляет собой логическое соединение устройств SMBus таким образом, что два устройства с одинаковым подчиненным адресом будут находиться в конфликте друг с другом. Арбитражная область является совокупностью одной или более физических шин, соединенных таким образом, что установка сигнала шины любым устройством на любой физической шине обнаруживается всеми устройствами на всех физических шинах в пределах арбитражной области [9, 16].

Способность к подключению шины

Соединение SMBus со слотами системной платы, предназначенными для плат расширения, является необязательным. Если интерфейс SMBus связан с одним слотом в монтажной панели, то он должен быть связан со всеми слотами в этом блоке. Слоты, соединенные с SMBus в блоке, должны находиться

в одном логическом сегменте SMBus. Данная логическая структура SMBus может быть разрушена во время инициализации системы, или "горячего" извлечения. Это позволяет системе определять, в каком PCI-слоте находится SMBus-устройство. Логическая структура должна быть восстановлена до окончания инициализации системы и во время нормальной работы. Типичная реализация одного физического сегмента SMBus показана на рис. 7.13, контроллер SMBus главной шины реализован системой [9, 16].

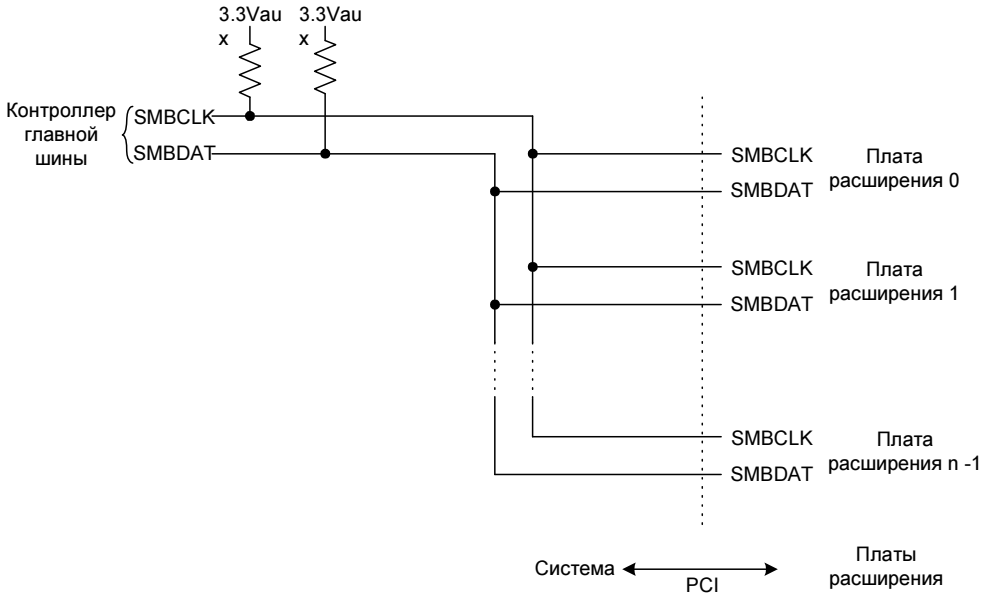


Рис. 7.13. Типичный физический сегмент SMBus

Блок SMBus не обязан поддерживать соединение с другими блоками. Данное требование не запрещает и не устраняет соединение с другими блоками, но не рассматривается как отдельная проблема. Конфигурация системной платы с одним или более сегментов шины PCI, связанных с интерфейсом SMBus, должна содержать один логический сегмент SMBus для всех сегментов шины PCI. Сегменты шины PCI никак не связаны с физическими шинами SMBus. Кроме этого, SMBus не соединяется с мостом подобно PCI-устройствам.

Поддержка "Master" и "Slave"

Все правила взаимодействия определены в спецификации SMBus 2.0, согласно этим правилам в системе существует два типа SMBus-устройств:

- ведущий (Master);
- ведомый (Slave).

Транзакции SMBus выполняются между ведущим и ведомым. Мастером является устройство, которое формирует команды, генерирует тактовый сигнал и завершает транзакцию шины. Ведомым является устройство, которое принимает или отвечает на транзакции шины. Системная плата, поддерживающая интерфейс SMBus, должна содержать мастера и ведомого и поддерживать арбитражный механизм SMBus.

Адресация и конфигурация

Протокол разрешения адреса или, иначе, *ARP* (Address Resolution Protocol) определен в спецификации SMBus 2.0 и используется для назначения адреса ведомого устройству SMBus. Системы, в которых интерфейс SMBus соединяется с PCI-слотами, должны реализовывать протокол ARP для назначения адресов ведомого устройствам SMBus на PCI-платах расширения. Система должна выполнить протокол ARP в логическом сегменте SMBus при выходе любого сегмента шины PCI, связанного с логическим сегментом SMBus, из состояния B3, или при выходе устройства в индивидуальном слоте, связанного с логическим сегментом SMBus, из состояния D3cold. Перед выполнением протокола ARP система должна вернуть все поддерживающие протокол ARP устройства с интерфейсом SMBus в их состояние по умолчанию. Допускается поддержка включения в блоке одной платы расширения с устройством SMBus, при этом адрес является фиксированным и находится в диапазоне C6h—C9h. Система с такой поддержкой не должна содержать устройства SMBus с фиксированным адресом в этом диапазоне и не должна назначать адреса из этого диапазона другим устройствам SMBus на PCI-платах расширения. При установке больше одной платы расширения могут возникнуть конфликты, решение которых не возлагается на систему.

Фиксированный адрес устройств

До выхода спецификации SMBus 2.0 механизм протокола ARP не был определен, и SMBus-устройства для ранних версий спецификации имели фиксированные адреса ведомых. Большинство систем использовали такие устройства только на системной плате. В связи с возникновением потребности в платах расширения с SMBus были введены несколько плат расширения для специфического применения. Эти платы расширения включали доступные SMBus-устройства с фиксированным адресом ведомого в диапазоне C6h—C9h. Спецификация версии 2.0 поддерживает использование таких плат расширения для обратной совместимости. Но предполагается, что все новые реализации будут использовать протокол ARP и таким образом избегать адресных конфликтов. Также допускается произвольная реализация механизма отображения устройств SMBus и физических слотов плат расширения, такой механизм должно выполнять конфигурационное ПО. ПО должно определить номер слота, в котором находится плата расширения, и связать полученный

номер с адресом SMBus-устройства на этой плате расширения. Во время выполнения отображения разрешается изолировать индивидуальные слоты, но по завершению система должна восстановить связи.

Электрические требования

Физические сегменты SMBus, которые соединены с разъемом PCI, должны быть реализованы на основе электрической спецификации постоянного тока, как определено в спецификации SMBus 2.0.

Максимальная емкостная нагрузка для физического сегмента SMBus составляет 400 пФ. Если физический сегмент SMBus включает PCI-слоты плат расширения, в расчетах должна использоваться максимальная емкость 40 пФ на каждую плату расширения. Абсолютное значение полного тока утечки для физического сегмента SMBus, источника, и/или потребителя энергии, не должно превышать 200 мкА, измеренное для напряжений $0{,}1V_{cc}$ и $0{,}9V_{cc}$.

Поведение SMBus при системном сбросе шины PCI

При подаче питания на SMBus-устройство оно должно выполнить инициализацию внутреннего состояния по умолчанию. Сигнал RST# не оказывает воздействия на логику SMBus-устройства. Это позволяет SMBus поддерживать связи при отсутствии таковых на шине PCI .

Требования платы расширения SMBus

Связь

Соединение с выводами SMBCLK и SMBDAT разрешено только для плат расширения PCI, которые поддерживают интерфейс SMBus с протоколом ARP. Любая плата расширения PCI, реализующая функциональные возможности SMBus, должна быть соединена с выводами SMBCLK и SMBDAT посредством PCI-разъема.

Поддержка "Master" и "Slave"

Плата расширения SMBus, предназначенная для ответа на команды или запросы данных от мастера, должна содержать функциональность ведомого и поддерживать протокол ARP. Поддержка функциональности мастера является необязательной, но при ее реализации плата расширения должна поддерживать арбитраж для мастеров.

Адресация и конфигурация

Плата расширения SMBus-устройства должна поддерживать протокол ARP для назначения ведомого адреса. Хотя в спецификации SMBus поддержка

протокола ARP является необязательной, в спецификации PCI данное требование обязательно для выполнения.

Питание

Интерфейсная логика SMBus-платы расширения может быть запитана от вывода 3.3Vaux на PCI-разъеме. Это обеспечивает доступность функций SMBus-платы расширения в то время, пока система находится в состоянии пониженного потребления питания. Такая плата расширения должна поддерживать вывод PME# из состояния D3cold, как определено в спецификации "PCI Power Management Interface". Интерфейсная логика SMBus может быть запитана от стандартного источника напряжения 3,3 В.

Электрические требования

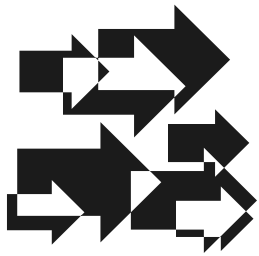
Устройства SMBus на платах расширения должны отвечать электрическим требованиям постоянного тока спецификации SMBus. Должно быть выполнено требование максимальной емкости 40 пФ на каждую плату расширения для каждого вывода сигнала SMBus. В этот предел 40 пФ входят:

- сумма емкостей нагрузок устройства;
- емкость длины линии от PCI-разъема платы расширения.

Абсолютное значение полного тока утечки источника, и/или потребителя энергии, не должно превышать 20 мкА, измеренное для напряжений $0{,}1V_{cc}$ и $0{,}9V_{cc}$.

Абсолютное значение суммы тока утечки, источника, и/или потребителя энергии для всех устройств интерфейса SMBus на плате расширения, не должно превышать 20 мкА, измеренное для напряжений $0{,}1V_{cc}$ и $0{,}9V_{cc}$. При выполнении этих требований количество SMBus-устройств на плате расширения не ограничено [9, 16].

ГЛАВА 8



Спецификация компонентов PCI

Введение

Разработка интегральных схем всегда тесно связана с вопросами сопряжения, оптимизации схемных решений и обеспечения надежного и согласованного функционирования элементов схемы. В общем случае к компонентам относятся устройства, выполненные в одной микросхеме и реализующие заданные функции. К ним могут относиться: операционные усилители, инверторы, преобразователи уровней, буферные схемы и другие логические элементы. Для выполнения требований сопряжения и согласованности спецификация PCI дает основные электрические требования и рекомендации для компонентов, предполагаемых к использованию в системах PCI. Данные требования нацелены на имеющиеся сегодня компоненты и материалы. Уменьшение размеров электронных компонентов влечет за собой уменьшение расстояния между цепями, а следовательно, возникает необходимость уменьшения напряжения питания для снижения наводок в цепях. Кроме того, уменьшение напряжения питания снижает потребляемую мощность. Именно этим объясняется постепенный переход от напряжения питания 5 В к 3,3 В. Для плавности перехода и обеспечения обратной совместимости спецификация поддерживает оба напряжения питания. В дальнейшем намечается переход на одно напряжение 3,3 В. В *главах 9* и *10* определены требования для системной платы и плат расширения. Разработчики стандарта стремились сделать соответствующие разделы спецификации независимыми, но подобное разделение практически труднодостижимо. Главы *8*, *9 и 10* тесно связаны между собой, и все производители системных плат, компонентов и плат расширения должны учитывать требования каждой из них.

Электрическая спецификация PCI предусматривает сигнальные среды 3,3 и 5 В. Они не должны быть спутаны с технологиями компонентов 3,3 и 5 В.

Компонент 3,3 В может быть разработан для работы в сигнальной среде 5 В, и наоборот; компонентные технологии могут быть смешаны в любой сигнальной среде. Сигнальные среды не могут быть смешаны; все компоненты на данной шине PCI должны использовать одинаковое соглашение о сигнальной среде: 3,3 или 5 В.

Приведенные требования предназначены не только для обеспечения электрической совместимости компонентов PCI, но и как фактически тестовые спецификации. Ответственность за комбинацию характеристик устройства и заводских испытаний, соответствующих приведенным здесь параметрам, лежит на разработчиках компонентов и специализированных интегральных схем. По умолчанию параметры компонента применяются к выводам микросхемы, а не к открытым контактным площадкам и не к разъемам платы расширения.

Выходные буферы PCI определены в терминах их вольт-амперных характеристик (ВАХ). Приемлемые пределы вольт-амперных характеристик обеспечиваются полным максимальным выходным сопротивлением и могут достигать приемлемого порогового напряжения в типичных конфигурациях, также они обеспечивают полное минимальное выходное сопротивление, которое удерживает параметры отраженной волны внутри допустимых пределов. Входы и выходы буфера имеют различные ВАХ. Эффективная буферная производительность определяется рабочей точкой переменного тока, которая определяет приемлемое пороговое напряжение вместе с требуемыми токами, чтобы достигнуть этого напряжения в типичных конфигурациях. Рабочая точка постоянного тока определяет условия установившегося режима, но в среде КМОП они являются минимальными и не указывают реальные параметры управления выводом. Затененные области на ВАХ (см. рис. 8.1 и 8.4) определяют допустимый диапазон для выходных характеристик. Параметры постоянного тока должны поддерживаться на определенном уровне в условиях установившегося режима. Параметры переменного тока должны гарантироваться для условий переходного переключения, которые могут представлять до 33% от тактового цикла. Знак на всех параметрах тока (направление истечения тока) направлен к земле внутри компонента; т. е. положительные токи текут в компонент, в то время как отрицательные токи вытекают из компонента.

Реализация энергосберегающих функций приводит к тому, что с шины и устройств будет сниматься питание. Дополнительный сигнал PME# имеет особое назначение, когда система находится в состоянии пониженного потребления. Использование этой линии не должно вызывать повреждение компонентов или каких-либо частей схемы, даже если не подключены выводы Vcc компонента [9, 21].

Сигнальная среда 5 В

Среда 5 В отвечает стандартам JEDEC.

Примечание

JEDEC — сокр. от Joint Electronic Device Engineering Council — объединенный совет по электронным устройствам.

Компоненты среды 5 В характеризуются большим значением потребляемой мощности и большими токами утечки. Характерными элементами среды 5 В являются серии ТТЛ-логики 74HC. В данном разделе определены требования для постоянного и переменного токов компонентов.

Спецификация по постоянному току

В данном разделе приведены параметры, при которых должен функционировать компонент в сигнальной среде 5 В. Постоянное напряжение обеспечивает работоспособность транзисторов в диапазоне их рабочих характеристик. В табл. 8.1 представлены параметры постоянного тока. Описываются напряжения питания, предельно допустимые параметры напряжений, требования к токам утечки, емкость и индуктивность выводов. Под *предельно допустимыми значениями* понимаются такие значения электрических величин, превышение которых может привести к выходу устройства из строя. Следует однозначно понимать, что приведенные значения параметров являются именно предельно допустимыми в течение короткого промежутка времени, работа устройства при таких значениях параметров не является нормой и недопустима в режиме нормальной эксплуатации. По отношению к транзисторам, предельно допустимыми значениями напряжений являются крайние значения рабочей транзисторной характеристики.

Таблица 8.1. *Спецификация по постоянному току для среды 5 В*

Символ	Параметр	Условия измерения	Мин.	Макс.	Ед. изм.	Прим.
V_{cc}	Поддерживаемое напряжение питания		4,75	5,25	В	
V_{ih}	Напряжение высокого логического уровня на входе		2,0	V_{cc} + 0,5	В	
V_{il}	Напряжение низкого логического уровня на входе		–0,5	0,8	В	
I_{ih}	Ток утечки высокого логического уровня сигнала	V_{ih} = 2,7 В		70	мкА	1

Таблица 8.1 *(окончание)*

Символ	Параметр	Условия измерения	Мин.	Макс.	Ед. изм.	Прим.
I_{il}	Ток утечки низкого логического уровня сигнала	V_{in} = 0,5 В		–70	мкА	1
V_{oh}	Напряжение высокого логического уровня на выходе	I_{out} = –2 мА	2,4		В	
V_{ol}	Напряжение низкого логического уровня на выходе	I_{out} = 3 мА, 6 мА		0,55	В	2
C_{in}	Емкость входного вывода			10	пФ	3
C_{clk}	Емкость вывода CLK		5	12	пФ	
C_{IDSEL}	Емкость вывода IDSEL			8	пФ	4
L_{pin}	Индуктивность вывода			20	нГн	5
I_{Off}	Ток утечки вывода PME#	V_o≤5,25 В V_{cc} откл.	—	1	мкА	6

Примечания

1. Входные токи утечки включают утечку тока высокоимпедансного вывода для всех двунаправленных буферов с тристабильными выводами.
2. Сигнальные линии без подтягивающих резисторов должны иметь выходной ток 3 мА. Линии, требующие "подтягивания", должны иметь ток 6 мА; к этим линиям относятся: FRAME#, TRDY#, IRDY#, DEVSEL#, STOP#, SERR#, PERR#, LOCK#, INTA#, INTB#, INTC#, INTD#, и для 64-разрядного режима: AD[63::32], C/BE[7::4]#, PAR64, REQ64# и ACK64#.
3. Абсолютная максимальная емкость контакта для PCI-входа — 10 пФ (исключая выводы CLK, SMBDAT и SMBCLK). Для компонентов, предназначенных к использованию только на системной плате, допускается емкость контакта до 16 пФ для обеспечения возможности использования типа корпуса PGA. Это означает, что для плат расширения должны использоваться различные типы корпусов компонентов для керамического PGA комплекта (т. е. PQFP, SGA и т. д.). Контактная емкость для выводов SMBCLK и SMBDAT не определена.
4. Более низкая емкость на этом входном выводе допускает не резистивную (т. е. прямую) связь с линиями AD[xx].
5. Рассматривается в качестве рекомендации, а не требования. Фактическое значение необходимо обеспечить спецификацией соответствующего компонента.
6. Это входной ток утечки — максимальный допустимый ток утечки вывода PME# драйвера с открытым коллектором, когда питание снято с вывода V_{cc} компонента. При этом предполагается отсутствие события, которое могло привести к установке устройством сигнала PME#.

Спецификация по переменному току

Свойства элементов электрической цепи описываются их статическими характеристиками. Приведенные в табл. 8.2 параметры по переменному току обобщаются в вольт-амперных характеристиках. Вольт-амперные характеристики транзистора содержат в себе информацию о его свойствах во всех режимах работы на больших и малых сигналах и о связях параметров между собой. По вольт-амперным характеристикам можно определить ряд параметров, не приводимых в справочной литературе, а также произвести расчеты цепей смещения, стабилизацию режима, оценку работы транзистора в широком диапазоне импульсных и постоянных токов, мощностей и напряжений [9, 23].

Скорость изменения выходного напряжения какой-либо логической схемы (как правило, схемы усиления, например, по отношению к операционным усилителям) не может быть мгновенной. Это обусловлено рядом причин: "компенсационной" емкостью, внутренними токами схемы; все это ограничивает скорость изменения выходного напряжения. Предельную скорость изменения выходного напряжения обычно называют *скоростью нарастания* [23].

Значения напряжений, соответствующих высоким и низким логическим уровням, могут колебаться в некотором диапазоне. Например, для высокоскоростной КМОП-логики (серии "HC") входные напряжения от уровня земли до 1,5 В представляются как низкий логический уровень, а напряжения питания +5 В — как высокий логический уровень. Следовательно, для переключающих напряжений ток должен изменяться в заданных пределах. Таким образом, под *током переключения высокого (или низкого) логического уровня* понимается предельно допустимое значение тока для заданного диапазона напряжений. Под *током замыкания высокого (низкого) уровня* понимается максимальный прямой ток через защитные диоды компонента [23].

Таблица 8.2. *Спецификация по переменному току для среды 5 В*

Символ	Параметр	Условия	Мин.	Макс.	Ед. изм.	Прим.
$I_{oh(AC)}$	Ток переключения высокого логического уровня	$0<V_{out}\le 1{,}4$	–44		мА	1
		$1{,}4<V_{out}<2{,}4$	$-44 + (V_{out} - 1.4)/0.024$		мА	1, 2
		$3{,}1<V_{out}<V_{cc}$		Фор. А		1, 3
	(контрольная точка)	$V_{out}= 3{,}1$		–142	мА	3

Таблица 8.2 *(окончание)*

Символ	Параметр	Условия	Мин.	Макс.	Ед. изм.	Прим.
$I_{ol(AC)}$	Ток переключения низкого логического уровня	$V_{out} \geq 2,2$	95		мА	1
		$2,2 > V_{out} > 0,55$	$V_{out}/0,023$		мА	1
		$0,71 > V_{out} > 0$		Фор. В		1, 3
	(контрольная точка)	$V_{out} = 0,71$		206	мА	3
I_{cl}	Ток замыкания низкого логического уровня	$-5 < V_{in} \leq -1$	$-25 + (V_{in} + 1)/0,015$		мА	
$slew_r$	Выходная скорость нарастания	0,4V до 2,4V	1	5	В/нс	4
$slew_f$	Выходная скорость понижения	2,4V до 04V	1	5	В/нс	4

Примечания и ограничения

1. ВАХ данных параметров иллюстрируется на рис. 8.1. Для значений токов переключения линий REQ# и GNT# допускаются величины равные одной второй из определенных здесь; т. е., для этих линий могут использоваться выходные драйверы с половиной уровня от размера данного уровня. Эти требования не применяются к линиям CLK и RST#, которые являются системными выводами. Параметр "ток переключения высокого уровня" не относится к выводам SERR#, PME#, INTA#, INTB#, INTC# и INTD#, которые являются выходами с открытыми коллекторами.
2. Данный сегмент кривой минимального тока выведен из рабочей точки AC и направлен к рабочей точке DC, а не к шине напряжения (как выполненный в понижающейся кривой). Это приращение предназначено, чтобы учесть дополнительный N-канал нагрузки.
3. Максимальные требования тока должны быть выполнены как "подтягивающее" напряжение драйверов свыше порогового напряжения. Уравнения, определяющие эти максимумы (A и B), обеспечиваются соответственными диаграммами на рис. 8.1. С целью упрощения испытания компонента, максимальная контрольная точка тока определена для каждой стороны выходного драйвера.
4. Этот параметр должен интерпретироваться как скорость накопления уровня через определенный диапазон, а не как мгновенная скорость в любой точке внутри диапазона перехода. Допускается дополнительная нагрузка этого выхода, но выход может быть и ненагруженным. Далее на рис. 8.2 изображен пример нагрузки выхода. Такая схема может применяться для выделе-

ния переднего и заднего фронта импульса. При этом расстояние между контактом и отверстием на плате не должно превышать ½ дюйма. Тем не менее, несмотря на рекомендательный характер, требуется соблюдение максимальных и минимальных параметров.

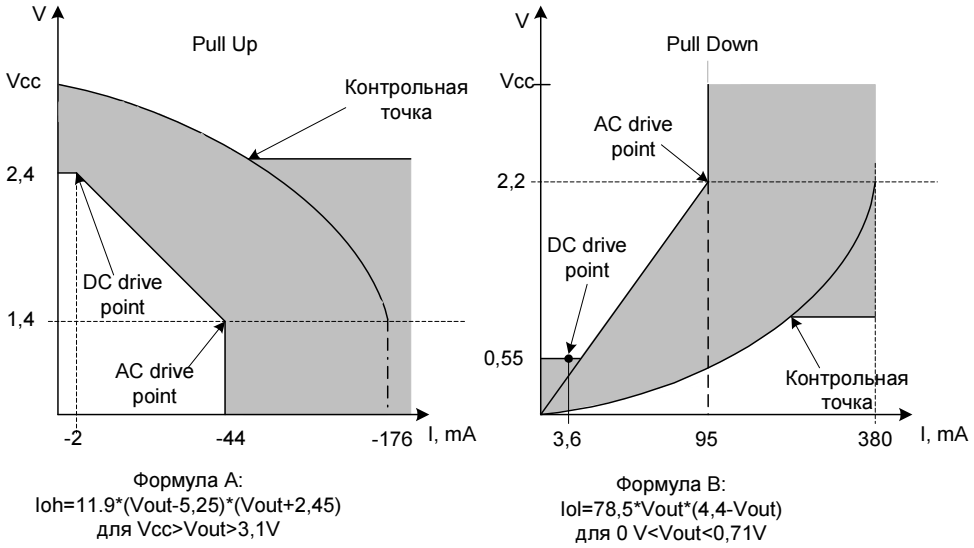


Рис. 8.1. BAX среды 5 В

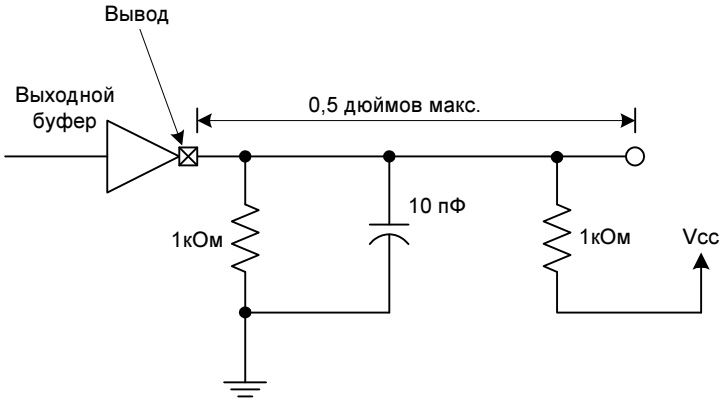


Рис. 8.2. Пример нагрузки выхода

Минимальные и максимальные модуляционные характеристики выходных PCI-буферов определяются вольт-амперными характеристиками (BAX), изо-

браженными на рис. 8.1. Эти ВАХ должны интерпретироваться как традиционные транзисторные характеристики по постоянному току со следующими исключениями: точка "DC drive point" является единственной позицией на кривых, в которых предназначается работа в установившемся режиме. Более высокие части кривых достигаются только на мгновение, пока на шине происходят переходные режимы. Точка "AC drive point" (реальное определение мощности буфера) определяет минимальную характеристику мгновенного значения тока, необходимую для переключения шины с одним отражением. Из статического, или установившегося, режима ток, связанный с точкой "AC drive point", должен достигнуть этого значения за выходное время задержки, T_{val}. Данное время задержки также включает необходимое логическое время. Распределение времени T_{val} на интервалы между тактами, логикой и буфером вывода не определено, но большее количество времени допускается для логической задержки внутри части. "Контрольная точка" определяет максимально допустимую характеристику мгновенного значения тока, чтобы ограничить шумы переключателя, и выбрана грубо на линии с нагрузкой 22 Ом.

Данные характеристики построены для наихудших условий. Минимальная характеристика подтягивания оценена для минимального значения V_{cc} и при высокой температуре. Минимальная характеристика понижения оценена для максимального значения V_{cc} и при высокой температуре. Максимальные контрольные точки кривой оценены для максимального значения V_{cc} и при низкой температуре. Входы должны быть заземлены. Соединение с шиной 5 В не требуется, но может быть необходимо для защиты устройств с напряжением 3,3 В. Заземление непосредственно на шину 3,3 В с простым диодом никогда не должно использоваться в среде 5 В. При использовании двойных шин питания могут возникнуть паразитные диодные пути от одной шины питания до другой. Эти диодные пути могут стать проводящими, если напряжение одной из шин питания на мгновение выйдет за пределы спецификации. Ограничительные диоды на шине питания, так же как и выходные подтягивающие резисторы, должны быть способны противостоять току короткого замыкания, пока драйверы находятся в третьем состоянии [21].

Максимальные значения переменного тока и защита устройств

В данном разделе приведена форма максимального сигнала переменного тока, а точнее форма максимально возможного значения напряжения переменного тока для среды 5 В. Под такой формой понимается форма сигнала при его предельных значениях по амплитуде и длительности переходного периода. Рассматриваемый в данном разделе пример является примером для наихудших эксплуатационных условий.

Рекомендуется, чтобы данные формы сигналов использовались как квалификационные критерии, т. е. как предельно допустимые, по сравнению с которыми оценивается надежность устройства на большом интервале времени. Данный пример не предназначен для использования в качестве заводских тестов; его основное предназначение состоит в том, чтобы определить уровень ошибкоустойчивости, который будет гарантироваться в соответствии с разработкой. Данный раздел охватывает только эксплуатационные условия для переменного тока и не затрагивает максимальные условия для постоянного тока. Среда PCI содержит множество реактивных элементов и в общем случае должна рассматриваться как среда незавершенной линией передачи. *Реактивными* называются нагрузки с чисто мнимым импедансом (идеальные конденсаторы и катушки). Эти элементы линейных цепей не поглощают энергии, а лишь частично запасают ее в электрическом или магнитном поле с последующей отдачей в электрическую цепь. На активной же нагрузке происходит однонаправленное преобразование электрической энергии в тепловую [9, 21].

Основное требование среды PCI состоит в том, чтобы сигнал отражался от конца линии и возвращался к драйверу до того, как сигнал считается переключенным. Таким образом, отраженный сигнал должен распространяться за время меньшее, чем время переключения. Как последствие особенностей этой среды, при некоторых состояниях драйверов, топологии устройств, импеданса системной платы, импеданса карты расширения и т. д., напряжение "разомкнутой цепи" на выводах PCI-устройств значительно превысит ожидаемое значение диапазона напряжения между землей и напряжением V_{cc}. Технология, используемая для реализации PCI, может измениться от производителя к производителю, так что нельзя принимать, что технология является естественно устойчивой к подобным эффектам. Такое превышение максимальных и минимальных значений напряжения обеспечивает синтез худшего случая среды переменного тока, для которой может быть оценена надежность устройства на большом интервале времени.

Все входы, двунаправленные выходы и тристабильные выходы, используемые в каждом PCI-устройстве, должны быть способны к продолжительному воздействию сигнала заданной формы. Тестовая форма сигнала обеспечивается эквивалентом источника напряжения с нулевым импедансом, и последовательно подключенными резисторами непосредственно к каждому входу или тристабильному выходному выводу PCI-устройства. Форма волны обеспечивается источником напряжения (или напряжением разомкнутой цепи) и величиной резистора, показанной на рис. 8.3. Форма волны разомкнутой цепи — составная часть моделируемого худшего случая[1]; резистор рассчитан,

[1] Форма сигнала взята для худших параметров драйвера, максимальной и минимальной системной конфигурации, без внутренних ограничительных диодов.

чтобы обеспечить худший случай для тока в эффективном (внутреннем) ограничительном диоде [9, 21].

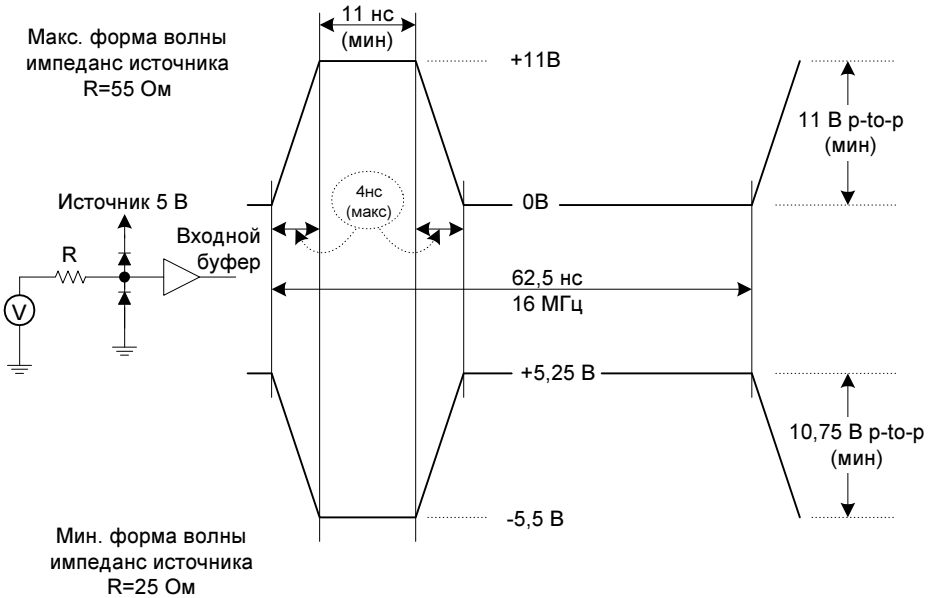


Рис. 8.3. Форма максимального сигнала для среды 5 В

Пояснения к диаграмме сигнала.

- Форма сигнала задается в резисторе, а не на выводе компонента (вывод обозначается как перечеркнутый квадрат).
- С эффективным заземлением форма сигнала на выводе микросхемы будет сильно изменена.
- Верхний ограничительный диод является необязательным, но при его использовании он должен быть связан с напряжением 5 В или линией $V_{I/O}$ платы расширения, но не с напряжением[2] 3,3 В.
- Для устройств, разработанных по технологии 3 В, верхний ограничительный диод на практике требуется для защиты устройства.
- Для ограничения переходных процессов в системах, которые имеют тенденцию выходить за установленные пределы, производители системных плат могут по желанию применять различные схемотехнические решения, чтобы понизить полное сопротивление цепи.

[2] Возможно использовать альтернативные ограничители, такие как диодный блок к шине 3,3 В или цепь к земле, если это гарантирует, что вывод I/O никогда не замкнется ниже уровня 5 В.

Сигнальная среда 3,3 В

Переход к среде 3,3 В связан с несколькими причинами. Уменьшение размеров компонентов требует снижения напряжения питания, с целью уменьшения возможных наводок между цепями. Среда 3,3 В соответствует стандарту JEDEC JESD8-B (Interface Standard for Nominal 3 V/3.3 V Supply Digital Integrated Circuits). Компоненты среды 3,3 В характеризуются меньшим потреблением и меньшими токами утечки. В данном разделе определены требования для постоянного и переменного токов [9, 21].

Спецификация по постоянному току

В данном разделе приведены параметры, при которых должен функционировать компонент в сигнальной среде 3,3 В. Постоянное напряжение обеспечивает работоспособность транзисторов в диапазоне их рабочих характеристик. В табл. 8.3 представлены параметры постоянного тока. Описываются напряжения питания, предельно допустимые параметры напряжений, требования к токам утечки, емкость и индуктивность выводов.

Таблица 8.3. *Спецификация по постоянному току для среды 3,3 В*

Символ	Параметр	Условия	Мин.	Макс.	Ед. изм.	Прим.
V_{cc}	Поддерживаемое напряжение		3,0	3,6	В	
V_{ih}	Напряжение высокого логического уровня на входе		$0,5V_{cc}$	$V_{cc} + 0,5$	В	
V_{il}	Напряжение низкого логического уровня на входе		–0,5	$0,3V_{cc}$	В	
V_{ipu}	Входное подтягивающее напряжение		$0,7V_{cc}$		В	1
I_{il}	Входной ток утечки	$0 < V_{in} < V_{cc}$		±10	мкА	2
V_{oh}	Напряжение высокого логического уровня на выходе	I_{out} = –500 мкА	$0,9V_{cc}$		В	
V_{ol}	Напряжение низкого логического уровня на выходе	I_{out}= 1500 мкА		$0,1V_{cc}$	В	

Таблица 8.3 *(окончание)*

Символ	Параметр	Условия	Мин.	Макс.	Ед. изм.	Прим.
C_{in}	Емкость входного вывода			10	пФ	3
C_{clk}	Емкость вывода CLK		5	12	пФ	
C_{IDSEL}	Емкость вывода IDSEL			8	пФ	4
L_{pin}	Индуктивность вывода			20	нГн	5
I_{Off}	Входная утечка PME#	$V_o \le 3{,}6$ В V_{cc} откл.		1	мкА	6

Примечания

1. Минимальное напряжение, для которого рассчитаны подтягивающие резисторы. Приложения, чувствительные к статическому использованию питания, должны гарантировать, что входной буфер проводит минимальный ток при этом входном напряжении.
2. Входные токи утечки включают утечку тока высокоимпедансного выхода (Hi-Z) для всех двунаправленных буферов с тристабильными выходами.
3. Абсолютная максимальная емкость вывода для входа PCI — 10 пФ (исключая выводы CLK, SMBDAT и SMBCLK). Для компонентов, предназначенных к использованию только на системной плате, допускается емкость контакта до 16 пФ для обеспечения возможности использования типа корпуса PGA. Это означает, что для плат расширения должны использоваться различные типы корпусов компонентов для керамического PGA комплекта (т. е. PQFP, SGA и т. д.). Контактная емкость для выводов SMBCLK и SMBDAT не определена.
4. Более низкая емкость на этом выводе необходима только для входа и учитывает нерезистивную связь с линиями AD[xx].
5. Рассматривается в качестве рекомендации, а не требования. Фактическое значение необходимо обеспечить спецификацией соответствующего компонента.
6. Это входной ток утечки — максимальный допустимый ток утечки вывода PME# драйвера с открытым коллектором, когда питание снято с вывода V_{cc} компонента. При этом предполагается отсутствие события, которое могло привести к установке устройством сигнала PME#.

Спецификация по переменному току

Параметры переменного тока сигнальной среды 3,3 В приведены в табл. 8.4. Определения вольт-амперных характеристик, скорости нарастания, тока пе-

реключения высокого (или низкого) логического уровня, тока замыкания высокого (низкого) логического уровня приведены в спецификации по переменному току для среды 5 В [9, 21].

Таблица 8.4. *Спецификация по переменному току для среды 3,3 В*

Символ	Параметр	Условия	Мин.	Макс.	Ед. изм.	Прим.
$I_{oh(AC)}$	Ток переключения высокого уровня	$0<V_{out}<0.3V_{cc}$	$-12V_{cc}$		мА	1
		$0.3V_{cc}<V_{out}<0.9V_{cc}$	$-17,1(V_{cc} - V_{out})$		мА	1
		$0.7V_{cc}<V_{out}<V_{cc}$		Фор. C		1, 2
	(контрольная точка)	$V_{out} = 0,7V_{cc}$		$-32V_{cc}$	мА	2
$I_{ol(AC)}$	Ток переключения низкого уровня	$V_{cc}>V_{out}>0,6V_{cc}$	$16V_{cc}$		мА	1
		$0,6V_{cc}>V_{out}>0,1V_{cc}$	$26,7V_{out}$		мА	1
		$0,18V_{cc}>V_{out}>0$			Фор. D	1, 2
	(контрольная точка)	$V_{out} = 0,18V_{cc}$		$38V_{cc}$	мА	2
I_{cl}	Ток замыкания низкого уровня	$-3<V_{in} \le -1$	$-25 + (V_{in} + 1)/0,015$		мА	
I_{ch}	Ток замыкания высокого уровня	$V_{cc}+4>V_{in} \ge V_{cc}+1$	$25 + +(V_{in} - V_{cc} - 1)/0,015$		мА	
$slew_r$	Выходная скорость нарастания	$0,2V_{cc} - 0,6V_{cc}$	1	4	В/нс	3
$slew_f$	Выходная скорость понижения	$0.6V_{cc} - 0,2V_{ccl}$	1	4	В/нс	3

Примечания

1. ВАХ данных параметров иллюстрируется на рис. 8.4. Значениям токов переключения для линий REQ# и GNT# допускаются величины равные одной второй из определенных здесь; т. е. для этих линий могут использоваться выходные драйверы с половиной размера данного уровня. Эти требования не применяются к линиям CLK и RST#, которые являются системными выводами. Параметр "ток переключения высокого уровня" не относится к выводам SERR#, PME#, INTA#, INTB#, INTC# и INTD#, которые являются выходами с открытыми коллекторами.

2. Максимальные требования тока должны быть выполнены как "подтягивающее" напряжение драйверов свыше порогового напряжения. Уравнения, определяющие эти максимумы (C и D), обеспечиваются соответственными диаграммами на рис. 8.4. С целью упрощения испытания компонента, максимальная контрольная точка тока определена для каждой стороны выходного драйвера.
3. Этот параметр должен интерпретироваться как скорость накопления уровня через определенный диапазон, а не как мгновенная скорость в любой точке внутри диапазона перехода. Допускается дополнительная нагрузка этого выхода, но выход может быть и ненагруженным. Далее на рис. 8.5 изображен пример нагрузки выхода. Такая схема может применяться для выделения переднего и заднего фронта импульса. При этом расстояние между контактом и отверстием на плате не должно превышать ½ дюйма. Тем не менее, несмотря на рекомендательный характер, требуется соблюдение максимальных и минимальных параметров [9, 21].

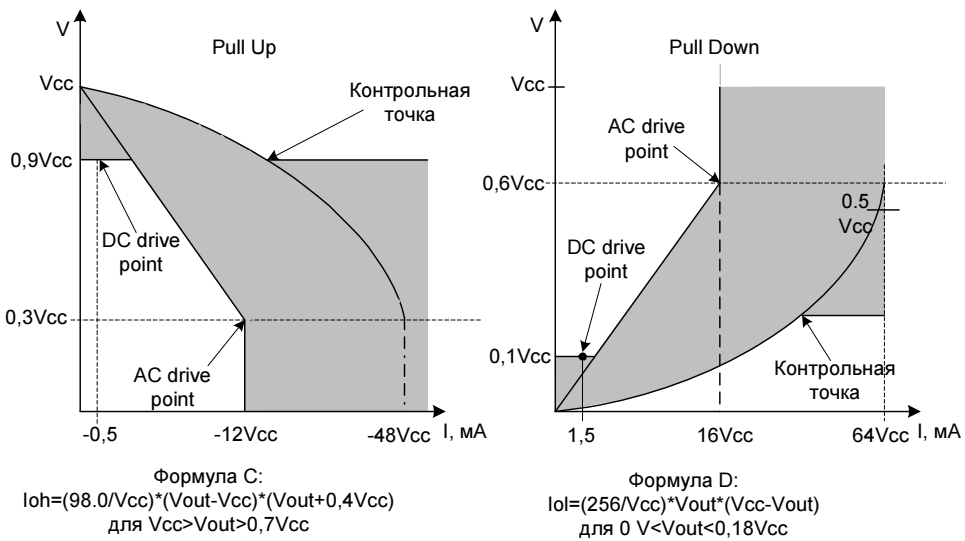


Рис. 8.4. BAX для сигнальной среды 3,3 В

Минимальные и максимальные модуляционные характеристики выходных PCI-буферов определяются вольт-амперными характеристиками (BAX), изображенными на рис. 8.4. Эти BAX должны интерпретироваться как традиционные транзисторные характеристики постоянного тока со следующими исключениями: точка "DC drive point" является единственной позицией на кривых, в которых предназначается работа в установившемся режиме. Более высокие части кривых достигаются только на мгновение, пока на шине происходят переходные процессы. Точка "AC drive point" (реальное определение мощности буфера) определяет минимальную характеристику мгновенного

значения тока, необходимую для переключения шины с одним отражением. Из статического, или установившегося, режима ток, связанный с точкой "AC drive point", должен достигнуть этого значения за выходное время задержки T_{val}. Данное время задержки также включает необходимое логическое время. Распределение времени T_{val} на интервалы между тактами, логикой и буфером вывода не определено, но большее количество времени допускается для внутренней задержки логики. "Контрольная точка" определяет максимальную допустимую характеристику мгновенного значения тока, чтобы ограничить шумы переключателя, и выбрана грубо на линии с нагрузкой 22 Ом [9, 21].

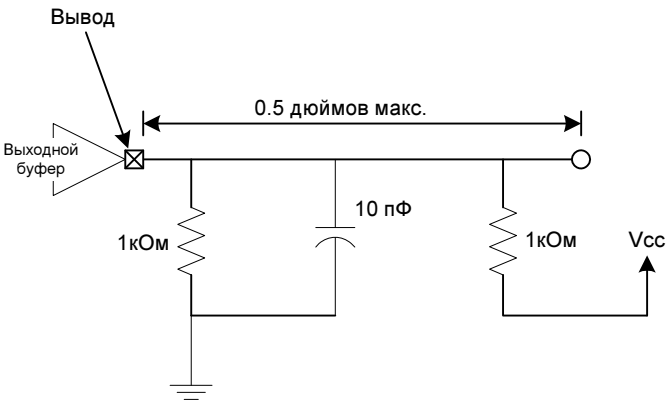


Рис. 8.5. Пример нагрузки выхода

Данные характеристики построены для наихудших условий. Минимальная характеристика подтягивающего напряжения оценена для максимального значения V_{cc} и высокой температуры. Минимум кривой понижающего напряжения оценен для максимального значения V_{cc} и высокой температуры. Максимальные контрольные точки кривой оценены для максимального значения V_{cc} и при низкой температуре.

Входы должны быть заземлены и соединены с линией Vcc напряжением 3,3 В. Когда используются двойные шины питания, могут существовать паразитные диодные пути от одного напряжения к другому. Эти диодные пути могут стать проводящими, если напряжение одной из шин питания выходит на мгновение из спецификации. Ограничительные диоды на шине питания, так же как выходные подтягивающие резисторы, должны быть способны противостоять току короткого замыкания, пока драйверы находятся в третьем состоянии.

Максимальные значения переменного тока и защита устройств

Смысл данного раздела совпадает с идентичным разделом для среды 5 В. Здесь также приведена форма максимального сигнала переменного тока для сигнальной среды 3,3 В. Рассматриваемый в данном разделе пример является примером для наихудших эксплуатационных условий. Под формой максимально возможного сигнала понимается форма при предельных его значениях по амплитуде и длительности переходного периода.

Все входные, двунаправленные и тристабильные выходы каждого PCI-устройства должны быть в состоянии выдержать продолжительное воздействие максимального сигнала. Предполагаемая схема содержит эквивалент источника напряжения с нулевым импедансом и управляется резистором, подключенным последовательно к каждому входному или тристабильному выходному выводу PCI-устройства. Форма сигнала, обеспечиваемая источником напряжения (или напряжением разомкнутой цепи), и значение резистора показаны на рис. 8.6. Форма сигнала разомкнутой цепи представляет совокупность моделируемых наихудших случаев. Резистор рассчитан так, чтобы обеспечить значение тока для подобного случая в эффективном (внутреннем) ограничительном диоде.

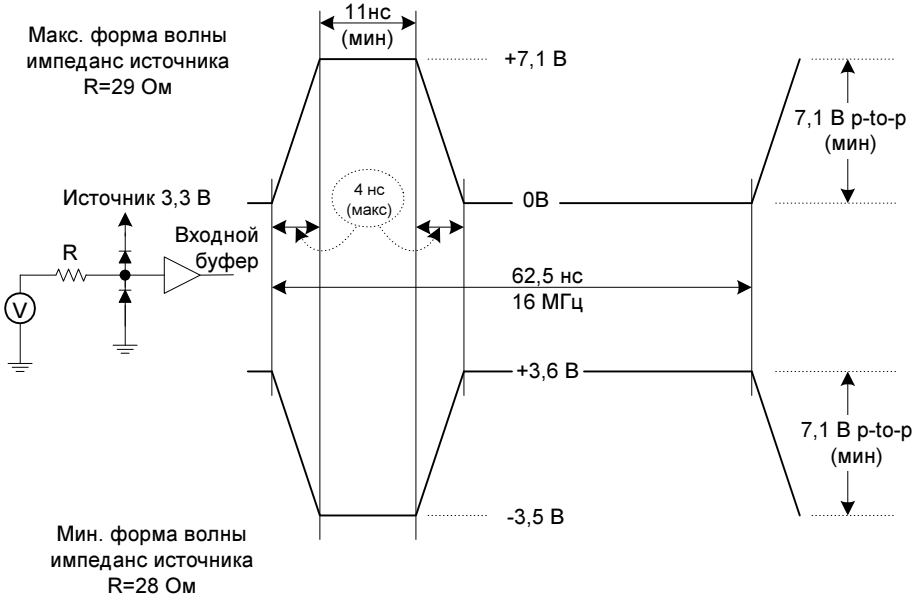


Рис. 8.6. Форма максимального сигнала для среды 3,3 В

Пояснения к диаграмме сигнала.

- Форма сигнала задается в резисторе, а не на выводе компонента (вывод обозначен как перечеркнутый квадрат).
- С эффективным заземлением амплитуда сигнала на выводе микросхемы будет сильно уменьшена.
- Для ограничения "звона" (переходный процесс в виде затухающих колебаний) в системах, которые имеют тенденцию выходить за установленные пределы, производители системной платы могут использовать различные схемотехнические решения, чтобы понизить полное сопротивление цепи.

Временные параметры

Шина PCI является синхронной шиной, все события происходят по сигналу синхроимпульса. Поэтому необходим стандарт синхроимпульса, которые используют все элементы шины. От полноты описания параметров синхросигнала зависит работа шины и устройств на ней. Поскольку скорость распространения в конечном счете ограничена и не существует идеальных элементов, то необходимо учесть переходные процессы. Переходные процессы невозможно устранить вообще, но возможно снизить до минимума [9].

Спецификация тактовых импульсов

Форма тактового сигнала должна быть определена. Тактовый сигнал представляет собой импульс трапециидальной формы, соответственно как и любой импульсный сигнал, он характеризуется различными параметрами: длительностью импульса, периодом, частотой, скоростью нарастания/снижения (или временем установления).

Схема может начать сбиваться, если состояние ее входа изменяется почти сразу же вслед за тактовым импульсом. Во избежание сбивания схемы для любого тактируемого устройства существует определенное "время установления" $t_{уст}$ и "время удержания" $t_{уд}$. Для того чтобы схема работала правильно, информация должна поступать на вход не позднее чем за время $t_{уст}$ до возникновения тактового перепада и оставаться неизменной по крайней мере в течение времени $t_{уд}$ после него.

Форма тактового сигнала как раз и определяется этими временными параметрами. Тактовый сигнал заданной формы или отвечающий заданным параметрам должен быть подан на соответствующий вход каждого PCI-компонента в системе. В случае плат расширения, соответствие спецификации по синхронизации относится к компоненту платы расширения, а не к слоту. На рис. 8.7 приведена форма тактового сигнала и требуемые точки измерения для сред передачи сигналов 5 и 3,3 В. Табл. 8.5 суммирует параметры тактового сиг-

нала. Таким образом, генератор тактовых сигналов должен генерировать подобный сигнал, либо любой другой сигнал при наличии схемы преобразования, а компоненты должны успевать отрабатывать в заданных временных интервалах.

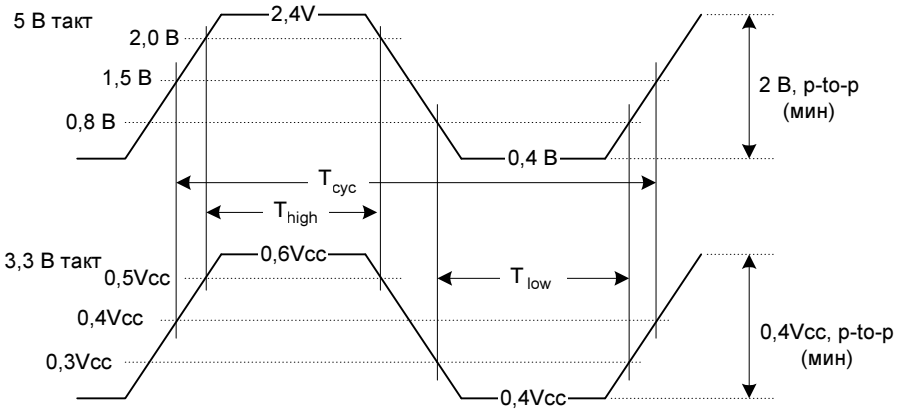


Рис. 8.7. Форма тактового сигнала PCI

Верхним уровнем тактового сигнала считается значение напряжения от 2,0 до 2,4 В для сигнальной среды 5 В и от $0,5V_{cc}$ до $0,6V_{cc}$ для среды 3,3 В. За нижний уровень тактового сигнала принимается значение между 0,4 и 0,8 В для среды 5 В и значение между $0,3V_{cc}$ и $0,2V_{cc}$. Амплитуда тактового импульса составляет 2 В для среды 5 В и $0,4V_{cc}$ для среды 3,3 В. Под временем нарастания понимается интервал, в течение которого сигнал нарастает от 10 до 90% своей мощности [9, 21].

***Таблица 8.5**. Спецификация синхроимпульса и сигнала сброса*

Символ	Параметр	Мин.	Макс.	Ед. изм.	Прим.
T_{cyc}	Период сигнала CLK	30	∞	нс	1
T_{high}	Время нахождения сигнала CLK в состоянии высокого логического уровня	11		нс	
T_{low}	Время нахождения сигнала CLK в состоянии низкого логического уровня	11		нс	
	Время нарастания сигнала CLK	1	4	В/нс	2
	Время нарастания сигнала RST#	50	—	мВ/нс	3

Примечания

1. Частота тактов может быть изменена в любое время во время функционирования системы, пока уровни тактов остаются "чистыми" (монотонными) и не превышают пределы минимального периода для сигнала CLK и нахождения его на верхнем и нижнем логическом уровне. Например, могут использоваться методы расширения спектра. Такт может быть остановлен только в состоянии низкого логического уровня. Компонентам, предназначенным для использования только на системной плате, разрешено не соблюдать временные требования для тактовых сигналов. Эти компоненты могут работать на любой одной частоте вплоть до 33 МГц.
2. Время нарастания и понижения определены в терминах скорости изменения уровня, измеряемого в В/нс. Эта скорость нарастания измеряется для всей амплитуды тактового сигнала.
3. Минимальная скорость нарастания сигнала RST# применяется к переднему фронту сигнала сброса и ее значение гарантирует, что шум системы, являющийся дрожанием в диапазоне переключения, не будет влиять на сигнал.

Временные параметры

Кроме основных параметров, влияющих на быстродействие, необходимо учитывать переходные процессы. Приведенные далее требования предопределяют параметры переходных процессов при синхронизации системы. Временные параметры устанавливают требования к устройствам, формирующим сигналы — генераторам (формирователям). Они не определяют временные параметры самих сигналов. Грубо говоря, приведенные здесь временные параметры определяют время реакции логики системных устройств, т. е. устройства должны успеть среагировать за установленные здесь интервалы. Они только косвенно определяют быстродействие системы. К временным параметрам относятся требования к выполнению тех или иных операций. Временные параметры определяются в зависимости от скорости нарастания и других параметров сигналов, их значения приведены в табл. 8.6. Также временные параметры определяют время распространения, таким образом, ограничивая длину линии. Некоторые обозначения требуют пояснения:

- T_{val} — задержка между сигналом CLK и установкой выходных сигналов на шине;
- $T_{val(ptp)}$ — задержка между сигналом CLK и установкой выходных двухточечных сигналов;
- T_{on} — задержка перехода сигнала из плавающего состояния в активное;
- T_{off} — задержка перехода сигнала из активного состояния в плавающее;
- T_{su} — время между установкой сигналов на входе и CLK на шине;
- $T_{su(ptp)}$ — время между установкой сигналов на входе и CLK для двухточечных сигналов;

- T_h — время удержания сигнала на входе в активном состоянии от начала активности CLK.

Таблица 8.6. *Временные параметры сигналов для сигнальной среды 3,3 и 5 В*

Символ	Параметр	Мин.	Макс.	Ед. изм.	Прим.
T_{val}	Задержка между CLK и шинными сигналами	2	11	нс	1, 2, 3
$T_{val(ptp)}$	Задержка между CLK и двухточечными сигналами	2	12	нс	1, 2, 3
T_{on}	Задержка перехода сигнала из плавающего состояния (Float) в активное состояние (Active)	2		нс	1, 7
T_{off}	Задержка перехода сигнала из активного состояния (Active) в плавающее (Float)		28	нс	1, 7
T_{su}	Время установления входа к CLK — для шинных сигналов	7		нс	3, 4, 8
$T_{su(ptp)}$	Время установления входа к CLK — для двухточечных сигналов	10, 12		нс	3, 4
T_h	Входное время удержания из CLK	0		нс	4
T_{rst}	Время активности сигнала RST# после подачи питания	1		мс	5
$T_{rst-clk}$	Время активности сигнала RST# после установления CLK	100		мкс	5
$T_{rst-off}$	Задержка перехода сигнала RST# из активного состояния в плавающее		40	нс	5, 6,7
T_{rrsu}	Время установки сигнала REQ64# после RST#	$10T_{cyc}$		нс	
T_{rrh}	Время удержания сигнала RST# после REQ64#	0	50	нс	
T_{rhfa}	Время между снятием сигнала RST# до первого конфигурационного доступа	2^{25}		такт	
T_{rhff}	Время между снятием сигнала RST# до первой установки сигнала FRAME#	5		такт	
T_{pvrh}	Подача питания после снятия сигнала RST#	100		мс	

Примечания

1. См. условия измерения выбора времени на рис. 8.8.
2. Для сигнальной среды 3,3 В:

   Минимальные значения временных параметров оценены с той же самой нагрузкой, что использовалась для измерения скорости нарастания (как показано в примечании 3 к табл. 8.4); максимальные значения временных параметров оценены с нагрузками цепи, для высокого и низкого логического уровней соответственно, передний и задний фронт соответственно.
3. Для сигнальной среды 5 В:

   Минимальные значения временных параметров оценены с эквивалентной емкостной нагрузкой 0 пФ; максимальные временные параметры оценены с эквивалентной емкостной нагрузкой 50 пФ. Фактическая тестовая емкость может изменяться, но результаты должны коррелировать с этими значениями. Более быстрые буферы могут вызывать небольшой дребезг, когда к ним привязана нагрузка емкостью 50 пФ. Эти помехи не будут оказывать какого-либо воздействия, если выходные буферы реализованы в полном соответствии со спецификацией скорости нарастания и кривой ВАХ.
4. Сигналы REQ# и GNT# являются двухточечными. Их временные параметры отличаются от шинных сигналов. GNT# имеет время установления 10 нс; REQ# — 12 нс. Все другие сигналы являются шинными.
5. См. условия измерения выбора времени в рис. 8.9.
6. Сигнал CLK является устойчивым, если он соответствует требованиям тактового сигнала. Сигнал RST# устанавливается и снимается асинхронно относительно CLK.
7. Все выходные драйверы должны находиться в плавающем состоянии во время активности сигнала RST#.
8. С целью измерения времени перехода сигнала из плавающего состояния в активное и обратно, считается, что вывод находится в высокоимпедансном состоянии или отключен, когда полный ток, проходящий через вывод компонента, меньше или равен величине тока утечки.
9. Время установления сигнала на входе применяется, когда устройство не управляет выводом. Устройства не могут управлять и принимать сигналы одновременно.

Условия испытаний и измерений

Компоненты должны соответствовать параметрам измерений, определенных в данном разделе. Условия испытаний и измерений являются несколько завышенными по сравнению с обычными условиями эксплуатации. Такие требования обеспечивают надежность функционирования и гарантию, что система будет работоспособна в критических условиях. Рис. 8.8, 8.9 и табл. 8.7 определяют условия, при которых сделаны временные измерения. Тестовые параметры компонентов заданы таким образом, что все временные параметры соответствуют минимальным скоростям нарастания такта (самый медлен-

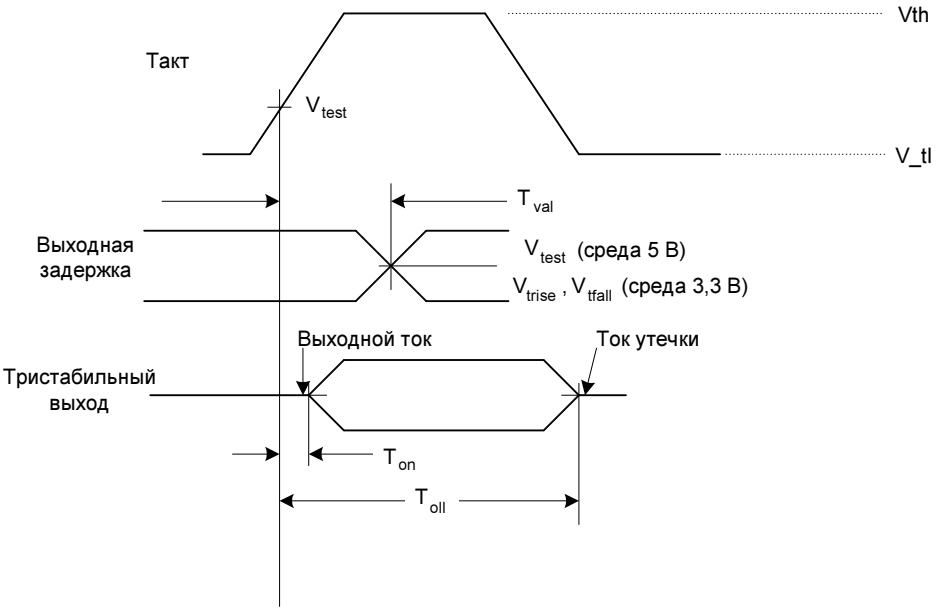


Рис. 8.8. Условия временных измерений выходного сигнала

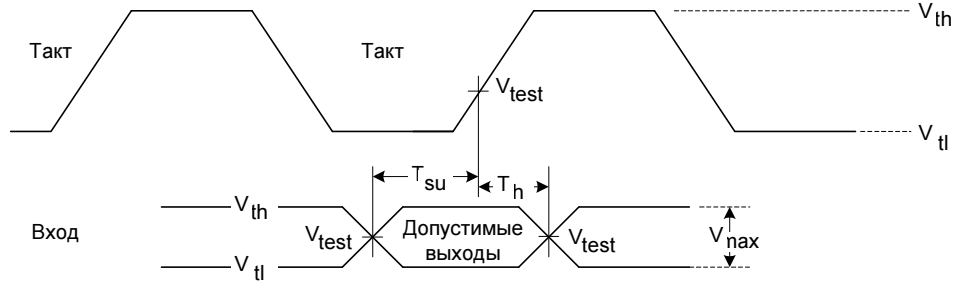


Рис. 8.9. Условия временных измерений входного сигнала

ный фронт) и амплитуде напряжения. При разработке необходимо гарантировать, что минимальные значения временных параметров также выполнены для максимальной скорости нарастания такта (самый быстрый фронт) и максимальной амплитуды напряжения. Кроме того, проект должен гарантировать надлежащую работу входа для входной амплитуды напряжения и скорости нарастания, значения которых превышают указанные тестовые условия. Через время T_{val} напряжение на выходе должно быть равно V_{test} для среды 5 В и V_{trise} для переднего и V_{tfall} для заднего фронта сигнала для среды 3,3 В. Напряжение на тристабильном выходе должно достигать номинального уровня

в пределах времени T_{on} и находиться на необходимом уровне до истечения интервала T_{off}. Все временные параметры измеряются от начала достижения тактового импульса значения напряжения V_{test}. Между установлением напряжения на входе и установлением напряжения сигнала CLK не должно пройти времени больше, чем T_{su}.

Параметры требуют пояснения:

- V_{th} — (V test high) тестовое напряжение высокого логического уровня тактового сигнала;
- V_{tl} — (V test low) тестовое напряжение низкого уровня тактового сигнала;
- V_{test} — тестовое напряжение, составляет половину от переднего или заднего фронта синхроимпульса;
- V_{trise} — напряжение переднего фронта сигнала;
- V_{tfall} — напряжение заднего фронта сигнала;
- V_{max} — максимальное значение амплитуды напряжения сигнала на входе.

Таблица 8.7. *Параметры измерения*

Символ	Сигнальная среда 3,3 В	Сигнальная среда 5 В	Ед. изм.
V_{th}	$0,6V_{cc}$	2,4	В (Прим.)
V_{tl}	$0,2V_{cc}$	0,4	В (Прим.)
V_{test}	$0,4V_{cc}$	1,5	В
V_{trise}	$0,285V_{cc}$		В
V_{tfall}	$0,615V_{cc}$		В
V_{max}	$0,4V_{cc}$	2,0	В (Прим.)
Скорость нарастания входного сигнала	1	1	В/нс

Примечание

Тестовое входное напряжение для среды передачи 3,3 В приведено с превышением на $0,1V_{cc}$; тестовое напряжение для сигнальной среды 5 В приведено с превышением 400 мВ (по V_{ih} и V_{il}). Временные параметры на практике не должны превышать эти значения. V_{max} определяет максимальную амплитуду сигнала, допустимую для измерения входного времени. Заводские испытания могут использовать различные значения напряжения, но результаты должны коррелировать с этими параметрами [9, 21].

Неопределенные входы и метастабильность

Время от времени различные входы могут находиться в неопределенном состоянии. Компоненты должны избегать логических эксплуатационных ошибок и метастабильности путем фиксирования входов только на заданных тактах. Синхронные сигналы должны быть действительным и определенным только на заданных тактах. Значения напряжений плавающих входов должны быть значительно смещены от области переключения, чтобы избежать логических, электрических и тепловых перегрузок. Невозможно избежать ситуаций, когда сигнал с малой скоростью нарастания (например, резистивное соединение с IDSEL) проходит через область переключения, совпадая с нарастанием тактового сигнала, но сигналы не должны оставаться в пороговой точке в течение многих периодов тактов. Существует множество условий, когда сигналы являются неопределенными. Их все необходимо учитывать в каждой разработке [9, 23].

Линии AD[31::00], C/BE[3::0]# и выводы PAR не определены в третьем состоянии для оборотного цикла шины. Такое состояние может продолжаться для нескольких циклов при ожидании ответа устройства в начале транзакции. Линия IDSEL не определена всегда, кроме циклов конфигурации. Если используется резистивная связь к адресной линии, то это может привести к нахождению плавающего вывода около области переключения большую часть времени. Выводы PME# и SERR# должны считаться неопределенными некоторое количество тактов после их снятия. Почти все сигналы будут находиться в неопределенном состоянии после установки сигнала RST#, а некоторые будут находиться в неопределенном состоянии в период времени после его снятия. Выводы с подтягивающими резисторами в конечном счете будут доведены до высокого логического уровня [9, 21].

Спецификация, обеспечиваемая производителем

Разработчики PCI-систем должны проводить электрическое моделирование шины PCI и компонентов для гарантии надлежащего функционирования. Производители компонентов должны предоставлять следующую информацию: (рекомендуется, чтобы эта информация была представлена в электронном виде в формате модели IBIS.)

- ❒ Значение емкости всех выводов.
- ❒ Значение индуктивности всех выводов.

- Выходные ВАХ при условиях переключения. Две кривые нужно дать для каждого используемого типа выхода: один для верхнего управления, другой для нижнего управления. На графиках должны быть представлены лучшая, нормальная и худшая кривые. Отклик вне пределов шины является критическим, так что диапазон напряжения должен охватывать от –3 до 7 В для сигнальной среды 3,3 В и от –5 до 10 В для среды 5 В.
- Входные ВАХ при условиях переключения. Входная ВАХ при условии тристабильного выхода также важна. На графиках также должны быть представлены лучшая, нормальная и худшая кривые в диапазоне от 0 до V_{cc}.
- Скорость нарастания/понижения уровня сигнала для каждого типа выхода.
- Все предельные характеристики, включая рабочую и нерабочую температуры, максимумы постоянного тока и т. д. [9].

Рекомендации по расположению выводов

В данном разделе приведены рекомендации по порядку расположения выводов PCI-компонентов. Поскольку количество контактов платы расширения ограничено, то необходимо минимизировать компоновку выводов. Это достигается, если разводка выводов компонента выровнена по отношению к разводке выводов платы расширения (разъема). Компоненты, предназначенные для использования только на системных платах, могут также выполнять такой порядок, чтобы добиться оптимального расположения. Рис. 8.10 показывает рекомендованную разводку выводов для типичного PCI-компонента в корпусе PQFP. Разводка точно выровнена в порядке сигналов относительно разъема платы расширения. Размещение и число выводов питания и земли зависит от устройства. Дополнительные сигналы, необходимые в версиях для 64-разрядной шины, продолжают отсчитываться вокруг компонента в направлении против часовой стрелки в том же самом порядке, в каком они появляются на расширении разъема на 64 бита [9].

Размещение вывода IDSEL как можно ближе к линиям AD[31::11] обеспечивает нерезистивную связь этого вывода с адресными линиями с небольшой дополнительной нагрузкой. Этот вывод имеет меньшее значение емкости, которая в некоторых случаях ограничит возможности его размещения [9].

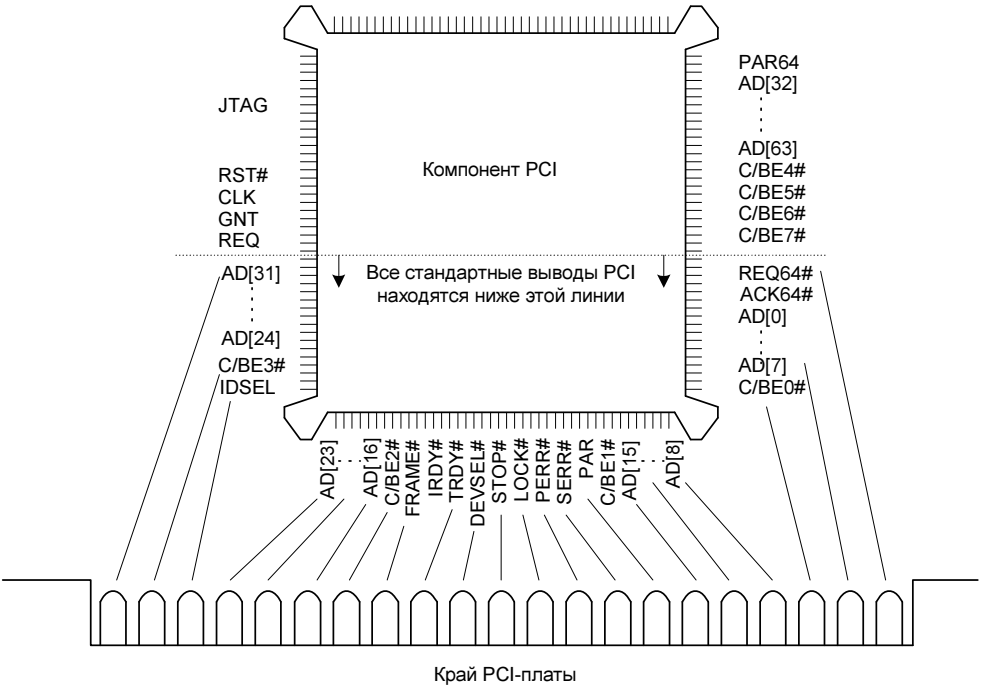
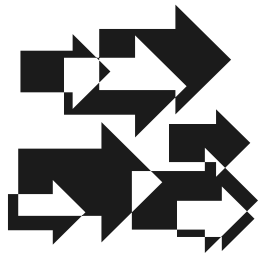


Рис. 8.10. Предполагаемое расположение выводов для компонентов в корпусе PQFP

ГЛАВА 9



Спецификация системной платы

Проектирование системной платы включает в себя несколько этапов, в том числе учет физических и электрических параметров, переходных процессов, перекоса синхронизации, распределения системного времени, требований сред распространения сигналов, требований к питанию и т. д. Эти параметры зависят не от компонентов, а именно от реализации конкретной системной платы, по отношению к компонентам они являются внешними. В прошлом, как правило, системная плата изготавливалась из фольгированного текстолита, схема нужной конфигурации получалась в результате травления медной фольги. Платы могут быть однослойными и многослойными. Все устройства должны тактироваться одним сигналом, следовательно, возможен перекос синхронизации из-за использования в качестве тактирующего сигнала с большим временем нарастания. При разработке схемы системной платы следует по возможности избегать запрещенных состояний системы; для предотвращения таких состояний необходимо использовать сигнал сброса, т. е. заложить на этапе проектирования возможность сброса всей системы. Для данного состояния надо определить параметры системы. Естественно, необходимо сразу спроектировать схему питания, для чего надо определиться, какие напряжения питания используются и в каких целях. Остальные параметры носят второстепенный характер, хотя их нужно учитывать: распределение системного времени, физические требования, импеданс, а также расстояния между слотами для плат расширения, назначения выводов слота [9].

Схемотехника системной платы

Перекос синхронизации

В системе присутствует общая синхронизация, задаваемая тактовым генератором. Тактовый генератор выполняется в виде отдельной микросхемы, рас-

полагающейся на материнской плате. Сигнал с данной микросхемы поступает в процессор, где он умножается на коэффициент. Далее тактовые импульсы передаются на все устройства системной платы, при этом устройства, такие как мосты, содержат схему преобразования частоты для взаимодействия с более медленными устройствами. Схема синхронизации налагает жесткие требования к форме и длительности тактовых импульсов. При взаимодействии устройств необходимо согласовывать по времени тактовый импульс, для этого надо определить максимально допустимое время рассогласования импульса. Выход за пределы этого времени может привести к потере данных или неправильной работе устройства или системы, а в худшем случае к выходу из строя устройства или системной платы [9].

Рассогласование по времени двух тактовых импульсов называется *перекосом синхронизации* или *перекосом тактовых импульсов*. Максимальный допустимый перекос синхронизации составляет 2 нс. Данное значение применяется не только в одиночной пороговой точке, но и во всех точках на уровне такта в диапазоне переключения, определенном в табл. 9.1 и рис. 9.1. Максимальный перекос измеряется между любыми двумя компонентами, а не между разъемами. Чтобы правильно оценить перекос такта, проектировщик системы должен принять во внимание распределение тактов на плате расширения.

Таблица 9.1. *Параметры перекоса синхронизации*

Символ	Сигнальная среда 3,3 В	Сигнальная среда 5 В	Ед. изм.
V_{test}	$0,4V_{cc}$	1,5	В
T_{skew}	2 (макс)	2 (макс)	нс

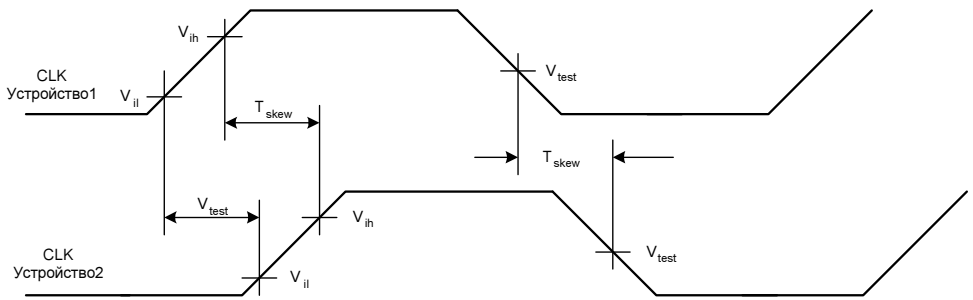


Рис. 9.1. Параметры перекоса тактовых импульсов

Системный сброс

При разработке и построении схемы необходимо по возможности избегать так называемых "мертвых" состояний. В общем случае под "мертвым" со-

стоянием понимается запрещенное состояние системы. В данное состояние система может перейти под воздействием каких-либо переходных помех по цепи питания. При разработке цифровых схем выявление запрещенных состояний является важной задачей, но решаемой не во всех случаях. Следовательно, наиболее простым и лежащим на поверхности решением является применение сигнала начальной установки, который мог бы возвращать систему в начальное состояние. Наличие данного сигнала решает проблему запрещенных состояний [9, 23].

Для шины PCI сигнал начальной установки определен и носит название сигнала сброса, обозначаемого RST#. Рассмотрим основные правила и требования при использовании данного сигнала.

Установка и снятие сигнала сброса RST# для шины PCI осуществляется асинхронно относительно сигнала CLK. Снятие сигнала RST# должно быть монотонно (свободно от дрожания) при прохождении через диапазон напряжений переключения входа (под входом понимается вход приемника) и должно отвечать минимуму скорости изменения напряжения T_{skew}, указанному в табл. 9.1. Спецификация PCI не запрещает реализацию синхронного сигнала RST#. Параметры синхронизации для сброса системы представлены в табл. 8.5 предыдущей главы, за исключением параметра T_{fail}. Данный параметр описывает реакцию системы на выход за пределы (отказ) спецификации одной или обеих шин питания. Если это происходит, паразитные диодные пути могут замкнуть накоротко активные выходные буферы. Следовательно, сигнал RST# устанавливается при сбое питания, чтобы перевести выходные буферы в неопределенное состояние.

Значение T_{fail} составляет минимум из следующих величин:

- 500 нс максимум для любой шины питания при выходе ее напряжения за пределы — превышении определенных допусков напряжения больше, чем на 500 мВ;
- 100 нс максимум для шины 5 В при падении ее напряжения ниже напряжения шины 3,3 В более чем на 300 мВ (т. е. когда напряжение шины 5 В станет меньше, чем 3 В).

Сигнал RST# должен быть выставлен во время включения питания или при сбое питания. При подаче питания сигнал RST# устанавливается за минимально возможное время. На рис. 9.2 показаны худшие условия при установке сигнала RST#. После того как сигнал RST# выставлен, PCI-компоненты должны асинхронно отключить (перевести в плавающее, неопределенное состояние) их выходы, но не реагировать на сигнал сброса, пока завершится время T_{rst} и $T_{rst\text{-}clk}$. Передний фронт сигнала RST# после подачи питания для любого устройства должен подаваться не раньше, чем через время T_{pvrh} после достижения определенных значений всех напряжений питания устройства.

Если сигнал RST# выставлен при соблюдении данного правила, т. е. все напряжения питания находятся внутри их определенных пределов, то минимальная длительность импульса RST# составляет время T_{rst}. На рис. 9.2 представлены временные параметры сигнала RST#. Система должна гарантировать, что шина будет находиться в состоянии "свободно" некоторое минимальное время задержки после снятия сигнала RST# с устройства до первой установки сигнала FRAME#. Это время задержки включено в табл. 8.6 как T_{rhff}. Устройству после снятия сигнала RST# может потребоваться больше времени, чем значение T_{rhff}. Конечный автомат устройства должен оставаться в состоянии сброса, пока не выполнятся следующие условия:

- сигнал RST# снят (находится в неактивном состоянии);
- устройство готово участвовать в протоколе сигнализации шины;
- шина находится в состоянии "свободно".

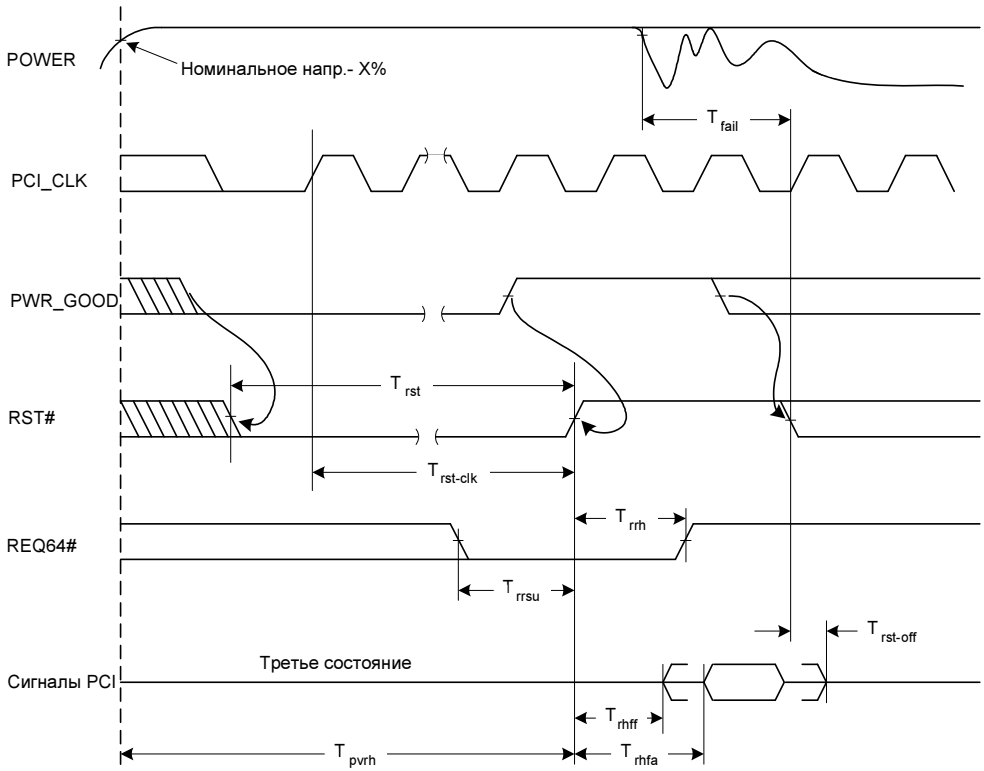


Рис. 9.2. Временные параметры сброса

Нарушение правил сброса

Рассмотрим случай превышения параметра T_{rhff} при соблюдении условий протокола сигнализации шины. В качестве примера рассмотрим устройство, работающее на частоте 66 МГц и содержащее схему ФАПЧ. Из-за наличия этой схемы может потребоваться больше времени, чем T_{rhff}, следовательно, может начаться передача между двумя другими устройствами. К этому времени закончится время ожидания, и на интерфейс устройства пойдут первичные такты, т. е. в середине передачи блока между двумя другими устройствами. При возникновении такой ситуации устройство, работающее на 66 МГц, должно проверять состояние шины перед предоставлением цели функции выбора.

Некоторые PCI-устройства должны быть готовы ответить как цель в пределах времени T_{rhff} после снятия сигнала RST#. Например, устройства, находящиеся между центральным процессором и загрузочной памятью (Flash-памятью с BIOS), должны быть готовы ответить как цель в пределах времени T_{rhff} после снятия сигнала RST#.

Все другие устройства должны быть готовы ответить как цель за время, не большее чем T_{rhfa} после снятия сигнала RST#. Рекомендуется, чтобы система ожидала, по крайней мере, период времени T_{rhfa} после снятия сигнала RST# с устройства перед выполнением первого доступа к этому устройству. Программное обеспечение, которое осуществляет доступ к устройствам до истечения времени T_{rhfa}, должно предусматривать возможность неготовности устройств (доступ, заканчивающийся с типом "Master-Abort") или ответ устройств "Retry" до истечения T_{rhfa}.

Рекомендуется выполнение инициализации и готовности устройств к приему транзакций (команда чтения конфигурации) за промежуток времени как можно меньший после снятия сигнала RST#. В некоторых случаях доступ может быть выполнен до истечения времени T_{rhfa}, при условии ранней готовности устройств. Например, система без PCI-слотов должна была бы ожидать в пределах требований инициализации встроенных устройств. Многофункциональное устройство, которое инициализирует ее собственные встроенные PCI-устройства, будет ожидать только в пределах временных требований инициализации этих устройств [9, 23].

Нагрузка

Линии шины PCI различаются по типу выходов: с открытым коллектором, тристабильные, подчиненные тристабильные и т. д. Характер и назначение этих линий определяют нагрузочные требования. Драйверы с тремя состояниями применяются для большинства линий PCI, но для некоторых примене-

ний необходимо использовать логику с открытым коллектором. К этим применениям относятся управление внешней нагрузкой, проводное ИЛИ и внешние шины. При использовании вентилей с открытым коллектором к источнику питания необходимо подключать внешний нагрузочный резистор. Таким образом, применительно к шине PCI, нагрузочные резисторы должны использоваться для выходов типа s/t/s, o/d.

Управляющие линии системной платы PCI должны подключаться через нагрузочные резисторы (подтягивающие к напряжению V_{cc} данной сигнальной среды), которые будут гарантировать стабильное значение напряжения, близкое к номинальному на этих линиях, когда шина не управляется ни одним из агентов. К этим сигналам также относятся FRAME#, TRDY#, IRDY#, DEVSEL#, STOP#, SERR#, PERR#, LOCK#, INTA#, INTB#, INTC#, INTD#, REQ64# и ACK64#. Двухточечные сигналы и разделяемые (общие) сигналы не требуют нагрузки — парковка шины гарантирует их стабильность. Системы, которые не поддерживают необязательный интерфейс SMBus, должны обеспечить отдельные подтягивающие резисторы (~5 кОм) на выводах SMBCLK и SMBDAT для разъемов системной платы. Системы, которые поддерживают интерфейс SMBus, должны обеспечить нагрузку (пассивную или активную) на выводах SMBCLK и SMBDAT. Нагрузки должны быть подключены к источнику питания, присоединенному к выводу 3.3Vaux разъема PCI для систем с дополнительным питанием, и к выводу +3.3V для систем без дополнительного источника [9, 21].

Формулы для расчетов минимума и максимума нагрузочных резисторов представлены далее. Величина R_{min} зависит от величины низкого выходного постоянного тока I_{ol}, а число нагрузок имеет только вторичное влияние. Наоборот, величина R_{max} управляется числом представленных нагрузок. Спецификация обеспечена для минимального значения R, которое вычислено для 16 нагрузок (это считается худшим случаем) и типичного значения R, которое вычислено как максимум значения R с 10-ю нагрузками. Максимальное значение R обеспечивается только формулой и будет наиболее высоким в системе с наименьшим числом нагрузок.

$$R_{min} = [V_{cc(max)} - V_{ol}] / [I_{ol} + (16 \times I_{il})], \quad (1)$$

где 16 — макс. количество нагрузок.

$$R_{max} = [V_{cc(min)} - V_x] / [num\_loads \times I_{ih}], \quad (2)$$

где V_x = 2,7 В для сигнальной среды 5 В, $V_x = 0{,}7V_{cc}$ для среды 3,3 В., num_loads — количество нагрузок.

Минимальные и типичные значения для обеих сигнальных сред приведены в табл. 9.2. Типичные значения резисторов были уменьшены на 10% от номинальных.

Таблица 9.2. *Минимальные и типичные значения нагрузочных резисторов*

Сигнальная среда	$R_{мин}$	$R_{тип}$	R_{max}
3,3 В	2,42 кОм	8.2 кОм	См. формулу 2
5 В	963 Ом	2.7 кОм	См. формулу 2

Центральный ресурс или любой компонент, содержащий арбитражный модуль, требуют небольшой нагрузки на каждом свободном выводе REQ# и каждом выводе REQ#, соединенным со слотом PCI. Таким образом, сигналы этих линий не будут находиться в плавающем состоянии. Величина нагрузки должна быть определена производителем центрального ресурса.

Системы, использующие сигнал PME#, должны обеспечить нагрузку на этом сигнале. Значение применяемого резистора рассчитывается с помощью указанных ранее формул (1 и 2), но при этом заменяя величину I_{ih} на I_{off} [9, 21].

Питание

Требования к питанию

Все PCI-разъемы требуют наличия четырех шин питания: +5 В, +3,3 В, +12 В и –12 В. Системы, содержащие PCI-разъемы, обязаны обеспечивать все четыре шины в каждой системе в соответствии с табл. 9.3. Системы могут произвольно реализовать дополнительное питание с вывода 3.3Vaux, как определено в спецификации "PCI Bus Power Management Interface". В системах без поддержки управления питанием шины PCI вывод 3.3Vaux должен рассматриваться как зарезервированный. Параметры тока в разъеме для двух 12 В шин приведены в табл. 9.3. Такие же ограничения для тока в разъеме шин 3,3 и 5 В отсутствуют и зависят от реализации. Необходимо учитывать, что плата расширения не должна потреблять больше 25 Вт со всех шин питания. Система обеспечивает необходимое питание для плат расширения, которое может быть распределено между разъемами произвольным способом. Выводы PRSNTn# в разъеме позволяют произвести оценку требований питания платы расширения, с целью определения потребляемой мощности. Допуски шин питания приведены в табл. 9.3. [9, 21].

Таблица 9.3. *Допуски шин питания*

Шина питания	Максимальный ток в разъеме, А
+3,3 В ± 0,3 В	7,6
+5 В ± 5%	5
+12 В ± 5%	0,5
–12 В ± 10%	0,1

Последовательность активизации шин питания

Не имеется никакой определенной последовательности, в которой должны активизироваться или деактивироваться четыре шины питания. Они могут включаться или выключаться в любом порядке. Система должна перевести в активное состояние сигнал RST# при включении питания и при выходе за пределы спецификации шин 5 и 3,3 В. Во время сброса все сигналы шины PCI доводятся до "безопасного" состояния [9].

Все шины питания должны быть развязаны с землей, чтобы обеспечить:

- ❒ разумное управление токами переключения;
- ❒ обратную цепь переменного тока для коротких импульсов.

Распределение временных параметров

При вычислении модели полной нагрузки шины PCI (или при вычислении общего количества PCI-нагрузок) особое внимание должно быть уделено максимальной длине линии и нагрузке плат расширения. Также, для плат расширения должна быть принята максимальная емкость контакта, равная 10 пФ, в то время как для устройств на системной плате допускается использование фактической емкости вывода.

Общий период синхронизации может быть разделен на четыре части. Допустимая выходная задержка (T_{val}) и входное время установки (T_{su}) определяется спецификацией компонента. Суммарный перекос синхронизации (T_{skew}) и максимальное время распространения по шине (T_{prop}) являются параметрами системной платы. Величина T_{prop} определена как 10 нс, но может быть увеличена до 11 нс за счет снижения величины перекоса синхронизации; т. е. сумма времен T_{prop} и T_{skew} вместе не должна превышать 12 нс, но ни при каких обстоятельствах время T_{skew} не может превышать 2 нс. Кроме того, на тактовых частотах меньше 33 МГц может быть построена система с большей топологией и значениями T_{prop}, превышающими определенные в спецификации PCI. Поскольку временные параметры (T_{val} и T_{su}) и перекос тактов электронных компонентов имеют фиксированные значения, то любое увеличение периода такта допускает эквивалентное увеличение величины T_{prop}. Например, на частоте 25 МГц (период тактового импульса 40 нс) время T_{prop} может быть увеличено до 20 нс. Данный компромисс относится только к системной плате, все реализации плат расширения должны предполагать работу на частоте 33 МГц. Правила измерения времени T_{prop} показаны на рис. 9.3 и 9.4 [9, 23].

Точка отсчета времени T_{prop} начинается в момент перехода сигнала выходного буфера через пороговую точку V_{trise} или V_{tfall} для сигнальной среды 3,3 В (V_{test} на рис. 9.4 для среды 5 В), при этом выход управляется определенной величиной $T_{val\ load}$.

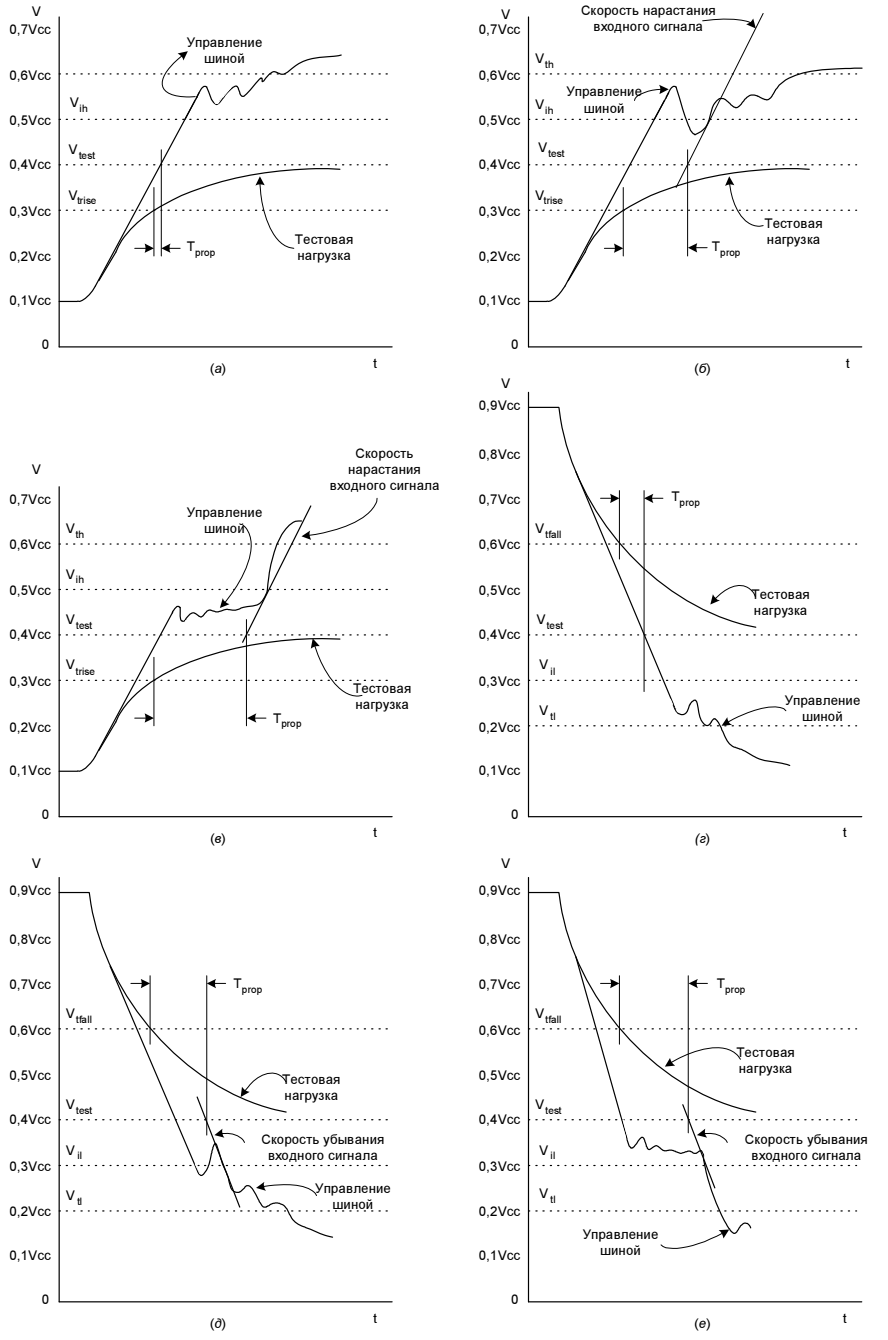


Рис. 9.3. Измерение T_{prop} для сигнальной среды 3,3 В

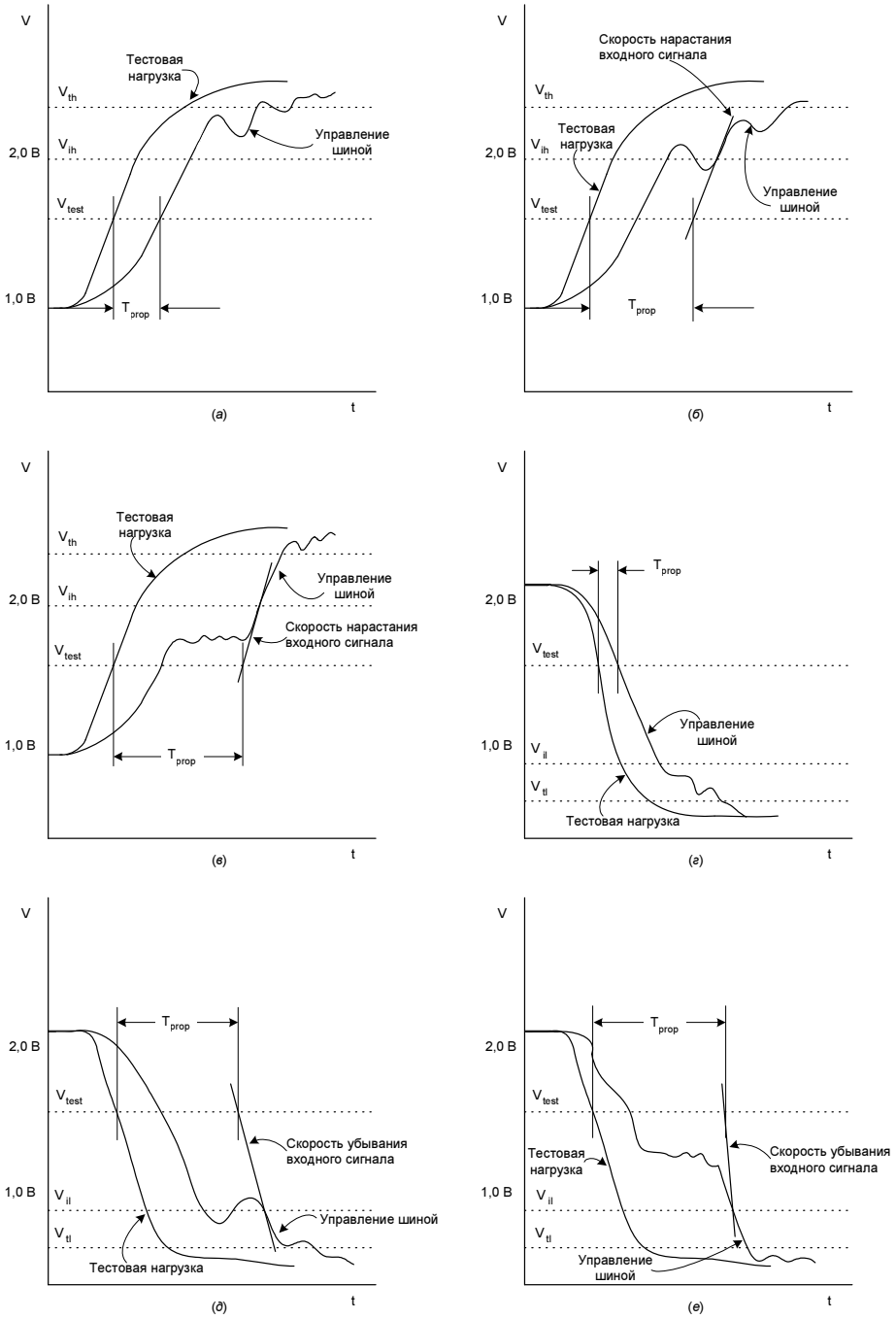


Рис. 9.4. Измерение T_{prop} для сигнальной среды 5 В

Конечная точка времени T_{prop} для любого входа определяется с помощью одного из двух методов измерения. Должен использоваться метод, который позволяет получить большее значение времени T_{prop}.

Метод 1. Окончанием T_{prop} считать время, когда сигнал на входе переходит через напряжение V_{test} в последний раз на рис. 9.3, *а*, *г* (для среды 5 В рис. 9.4, *а*, *г*).

Метод 2. Строится линия, наклон которой равен скорости нарастания входного сигнала из табл. 8.7 пересекающая точку, где сигнал на входе пересекает V_{ih} (повышающийся) или V_{il} (понижающийся) в последний раз. Окончанием T_{prop} является время, когда построенная линия пересекает V_{test} на рис. 9.3, *б*, *в*, *д*, *е* (рис. 9.4, *б*, *в*, *д*, *е* для среды 5 В).

Определение конечной точки времени T_{prop}

Расчет конечной точки времени T_{prop} всегда определяется методом, в результате которого получается большее значение T_{prop}. Значение определяется формой сигнала во входном буфере. Когда уровень сигнала повышается или понижается от напряжения V_{test} до последнего пересечения с напряжением V_{ih} или V_{il} со скоростью, большей чем скорость нарастания входного сигнала из табл. 8.7, метод 1 даст большую величину, как это показано на рис. 9.3, *а*, *г* для сигнальной среды 3,3 В (на рис. 9.4, *а*, *г* для среды 5 В).

Когда уровень сигнала повышается или понижается от напряжения V_{test} до последнего пересечения с напряжением V_{ih} или V_{il} со скоростью, меньшей чем скорость нарастания входного сигнала, то большую величину даст метод 2. Другими словами, если уровень сигнала после пересечения V_{test} изменяется медленно или сильно колеблется (дребезжит) между V_{ih} или V_{il}, метод 2 даст большую величину, как это показано на рис. 9.3, *б*, *в*, *д*, *е* для среды 3,3 В (на рис. 9.4, *б*, *в*, *д*, *е* для среды 5 В) [9].

Значения параметров рисунков 9.3 и 9.4 для частоты 33 МГц приведены в табл. 8.1, 8.3 и 8.7. При определении значения T_{prop} худшим случаем является установление сигнала во всех других устройствах на шине. Параметр T_{prop} не влияет на время установления сигнала на выходе драйвера, т. к. устройствам не разрешено управлять и принимать сигнал одновременно. Корректное распространение сигнала PCI достигается за счет диодов, входящих в состав компонентов для ограничения отражений и выполнения требований времени T_{prop}. В незавершенных линиях передачи при значительном расстоянии от PCI-компонента следует добавить активную нагрузку (например, диод) в ненагруженный конец шины, чтобы гарантировать адекватное качество сигнала. Поскольку протокол передачи сигналов зависит от начального отражения, то пассивная нагрузка не подходит [9, 23].

Физические требования

Четырехслойные системные платы

Распределение выводов питания предусматривает возможность применения четырехслойных системных плат. Разрешено применять "расщепление плоскости питания", т. е. создавать участок с напряжением питания 3,3 В в плоскости с питанием 5 В, который соединен со всеми выводами разъема 3,3 В. Хотя это стандартная технология, распространение сигналов непосредственно по этому расщеплению плоскости может привести к проблемам целостности сигнала. Расщепление в плоскости разрушает обратный путь переменного тока для сигнала, создавая неоднородность импеданса. Рекомендованное решение состоит в распределении сигналов высокой частоты по различным плоскостям. Сигнальные линии должны полностью находиться или в плоскости 3,3 В, или в плоскости 5 В. Линии, которые необходимо провести через разные плоскости, должны быть направлены на противоположной стороне системной платы таким образом, чтобы они были связаны с нерасщепленной плоскостью земли. Если это невозможно, и линии должны пересечь расщепленную плоскость, эти две плоскости должны иметь емкостную связь порядка 0,01 мкФ в виде быстродействующих конденсаторов для каждых четырех сигналов, пересекающих плоскость. Конденсатор должен быть размещен на расстоянии, не больше чем 0,25 дюймов от точки пересечения [9, 23].

Импеданс системной платы

На импеданс системной платы не налагается ограничение или максимальный предел. Но при разработке системной платы должны быть учтены следующие параметры:

- ❒ длительность и частота сигнала должны обеспечивать полное время распространения сигнала в обоих направлениях на шине в пределах указанной задержки распространения 10 нс;
- ❒ значение нагруженного импеданса в любой точке управления должно быть таким, чтобы выход PCI-устройства (определяемый его BAX) мог соответствовать требованиям входов устройств с одним отражением. Это включает нагрузки плат расширения.

Рабочая частота может попеременно изменяться для дополнительного распространения сигнала на шине. Это необходимо при реализации систем, которые не в состоянии выполнить данные два ограничения. Утверждение рассматривается в качестве рекомендации [9, 21].

Назначение выводов разъема

Разъем PCI содержит все линии, определенные для компонентов, плюс два дополнительных вывода. Системные платы должны разделять эти выводы для емкостной связи с быстродействующими конденсаторами 0,01 мкФ, потому что один или оба вывода обеспечивают обратный контур переменного тока. Эти выводы не могут быть соединены с шиной или как-либо иначе соединяться друг с другом на системной плате. Если системная плата использует эти выводы для получения информации о требованиях питания платы расширения, то каждый из них на системной плате должен быть нагружен согласующим резистором (сопротивлением приблизительно 5 кОм). Зарезервированные выводы должны быть свободными во всех разъемах. Специальный вывод 38B (вывод № 38 в ряду B) имеет логическое значение в слотах, совместимых со спецификацией PCI-X. Для обычных PCI-разъемов этот вывод должен рассматриваться как стандартный контакт земли (GND); т. е. этот вывод должен быть связан с плоскостью земли. Специальный вывод 49B — имеет логическое значение в слотах совместимых с частотой 66 МГц (M66EN), где он должен отдельно соединяться с шиной и иметь определенную нагрузку. Для всех других PCI-разъемов этот вывод нужно рассматривать как стандартный контакт земли (GND); т. е. вывод разъема должен быть связан с плоскостью земли. Выводы "+3.3V (I/O)" и "+5V (I/O)" являются специальными выводами питания для определения и управления сигнальными шинами на универсальных платах расширения. На системной плате эти выводы соединяются с плоскостью 3,3 или 5 В [9, 21].

Реализация системной платы

Платами расширения поддерживаются два типа системной платы PCI:

- ❒ слоты платы расширения, установленные на системной плате,
- ❒ слоты плат расширения, установленные на *riser-плате* (расширительной плате).

На рис. 9.5 показано размещение слотов на системной плате. Рисунок показывает размещение слотов 3,3 и 5 В относительно друг друга. Слоты 3,3 и 5 В показывают размерную информацию. Допуски должны быть совместимы с рекомендациями производителей слотов [9].

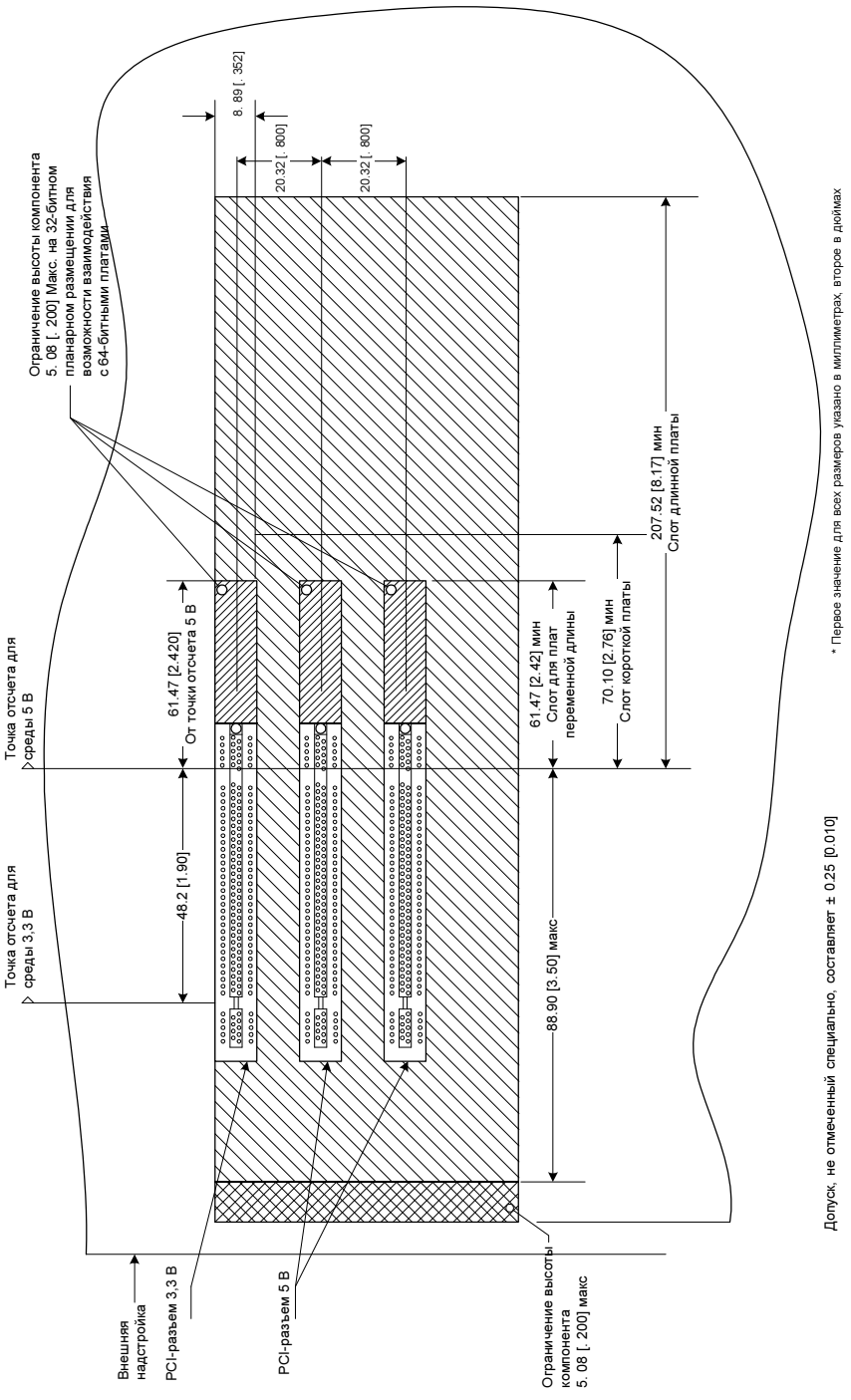
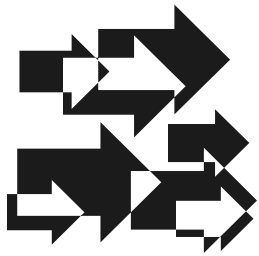


Рис. 9.5. Размещение слотов на системной плате

ГЛАВА 10



Спецификация плат расширения

Назначение выводов плат расширения

Разъем PCI содержит все линии, определенные для PCI-компонентов, плюс две дополнительные линии PRSNT1# и PRSNT2#. Данные линии предназначены для указания физического наличия платы расширения в слоте и предоставлении полной информации о требованиях к питанию платы расширения. В табл. 10.1 представлено соответствие состояния сигналов и конфигурации платы расширения.

Таблица 10.1. *Параметры питания платы расширения*

PRSNT1#	PRSNT2#	Конфигурация платы расширения
Сигнал	Сигнал	Плата расширения отсутствует
Земля	Сигнал	Плата расширения присутствует, 25 Вт макс.
Сигнал	Земля	Плата расширения присутствует, 15 Вт макс.
Земля	Земля	Плата расширения присутствует, 7,6 Вт макс.

При указании величины потребления питания должно быть показано полное максимальное значение для платы расширения, включая все питающие напряжения. Платы расширения могут произвольно запитываться от шин 3,3 или 5 В. Кроме того, если плата расширения имеет дополнительные гнезда для памяти или иные разъемы для дополнительных устройств, то должно быть указано полное питание с учетом дополнительных устройств. Платы расширения, не поддерживающие периферийное сканирование JTAG, должны реализовывать линии TDI и TDO (выводы 4A и 4B), чтобы не нарушалась цепь сканирования. Специальный вывод 38B имеет логическое значение в PCI-X совместимых платах расширения. Для остальных плат расширения PCI

этот вывод должен рассматриваться как стандартный контакт земли (GND); т. е. должен быть связан с землей платы расширения. Специальный вывод 49B имеет логическое значение для плат расширения, работающих на частоте 66 МГц, и должен быть соединен определенным способом. Для остальных плат расширения этот вывод должен рассматриваться как стандартный вывод земли.

Выводы +VI/O являются специальными контактами питания для того, чтобы определить и управлять шинами PCI на универсальной плате расширения. На этой плате расширения буферы ввода-вывода PCI-компонента должны запитываться только от этих выводов питания [9].

Питание

Емкостная развязка

В типичной реализации плат расширения емкость между плоскостью питания V_{cc} и плоскостью земли обеспечит нормальную развязку для выводов V_{cc} разъема. Максимальная длина линии от контактной площадки разъема до плоскости V_{cc}/GND составляет 0,25 дюймов. На универсальной плате расширения шина питания буфера ввода-вывода, вероятно, не будет иметь емкости для плоскости земли, чтобы обеспечить необходимую развязку. Выводы +VI/O должны быть развязаны с землей конденсатором со средним значением емкости 0,047 мкФ на каждый контакт. Все выводы питания +3.3V и любые неиспользованные выводы +5V, а также VI/O разъема PCI обеспечивают обратный контур переменного тока и должны быть соединены с плоскостью земли на плате расширения. При этом необходимо выполнять следующие условия, для гарантии их функционирования в качестве эффективных контрольных точек переменного тока:

- емкостная развязка должна составлять в среднем 0,01 мкФ для каждого вывода V_{cc};
- длина линии от контактной площадки вывода до контактной площадки конденсатора не должна превышать 0,25 дюймов, при ширине линии не больше 0,02 дюймов;
- конденсатор может быть общим для неограниченного количества выводов при условии выполнения двух первых условий.

Примечание

Обратите внимание, что здесь наименования выводов (шин, линий) питания английское, в тексте ранее часто использовались русские наименования, например, "+3,3 В", что соответствует "+3.3V". В действительности на платах обозначение именно английское, а в тексте книги, где используется обозначение "+3,3 В", имеется в виду все-таки вывод "+3.3V" с напряжением +3,3 В.

Максимальная потребляемая мощность

Максимальная потребляемая мощность для платы расширения любого типа составляет 25 ватт и включает в себя полное потребление со всех шин питания разъема (+3.3V, +5V, +VI/O, +12V, –12V, +3.3Vaux). Полная расходуемая мощность может обеспечиваться с любой шины питания. Большинство плат расширения будет расходовать значительно меньше 25 Вт. По этой причине рекомендуется, чтобы платы расширения, расходующие больше 10 Вт, использовали возможности перевода в состояние пониженного потребления питания, чтобы расходовать 10 Вт или даже меньше. Находясь в этом состоянии, плата расширения должна обеспечить полный доступ к ее пространству конфигурации и должна выполнять требуемые функции начальной загрузки, например, текстовый режим графического адаптера. Выполнение всех остальных функций платы расширения может быть приостановлено по необходимости. Режим экономичного питания достигают различными путями:

- уменьшением тактовой частоты платы расширения, что уменьшает производительность, но не ограничивает функциональные возможности;
- некоторые функциональные блоки могут быть отключены при помощи полевого транзистора (КМОП-транзистора), что может ограничить функциональные возможности.

После инициализации драйвера платы расширения он может перевести плату в полнофункциональное состояние, используя аппаратно-зависимый механизм выбора (возможно на основе информации из регистра). В системах с расширенным управлением питания драйвер устройства должен сообщить о потребляемой мощности устройства до полного включения платы расширения. Это позволит системе определить, в состоянии ли она обеспечить питание для всех плат расширения в текущей конфигурации.

Плата расширения ни при каких условиях не может быть самостоятельным источником питания для системной платы. Единственным исключением является специальная разработка платы расширения для обеспечения питания данной системы. В некоторых случаях платы расширения, способные к передаче сигналов в среде 3,3 В, могут иметь различные механизмы, которые косвенно являются источником питания по отношению к системе (системной плате), и поэтому названное требование будет нарушено. Данные особенности необходимо контролировать, например, плата расширения, содержащая компоненты, соединенные с шиной 3,3 В, может создать "насос зарядов", который направляет избыток энергии переключения шины назад в системную плату. Наоборот, выходные буферы ввода-вывода, функционирующие на шине 3,3 В, но используемые в сигнальной среде 5 В, могут выделять избыток зарядов с шины в сеть питания 3,3 В после управления шиной 5 В. "Неумышленный" источник питания с любым таким механизмом должен управ-

ляться надлежащей развязкой и достаточной локальной нагрузкой по цепи питания, чтобы рассеять любое питание, "произведенное" на плате расширения. Это требование не относится к шуму, генерируемому на шине питания, пока среднее значение шумовых колебаний в сети постоянного тока, за любые два периода тактов, равно нулю [9, 21].

Параметры среды распространения

Предельные значения для длины проводников на печатных платах

Установлены следующие предельные ограничения длины печатных проводников между разъемом печатной платы и микросхемой устройства PCI:

- максимальная длина проводника для всех линий 32-битного интерфейса составляет 1,5 дюйма для 64- и 32-битных плат расширения. К данным линиям относятся все группы, за исключением системных линий, линий прерывания, линий поддержки SMBus и линий периферийного сканирования;
- максимальная длина проводника дополнительных линий для 64-битного расширения составляет 2 дюйма для всех 64-битных плат расширения;
- максимальная длина проводника линии CLK составляет 2,5 ± 0,1 дюйма для 64- и 32-битных плат расширения, использование данной линии допускает только одну нагрузку.

Четырехслойные платы расширения

Требования и рекомендации по разработке четырехслойных плат расширения повторяют требования для четырехслойных системных плат. Распределение выводов питания предусматривает возможность применения четырехслойных системных плат. Разрешено использовать "расщепление плоскости питания". Хотя это стандартная технология, распространение сигналов непосредственно по этому расщеплению плоскости может привести к проблемам целостности сигнала. Расщепление в плоскости разрушает обратный контур переменного тока для сигнала, создавая неоднородность импеданса. Рекомендованное решение состоит в распределении сигналов высокой частоты по различным плоскостям. Сигнальные линии должны полностью находиться или в плоскости 3,3 В, или в плоскости 5 В. Линии, которые необходимо провести через разные плоскости, должны быть направлены на противоположной стороне системной платы таким образом, чтобы они были связаны с нерасщепленной плоскостью земли. Если это невозможно и линии должны пересечь расщепленную плоскость, то эти две плоскости должны иметь

емкостную связь порядка 0,01 мкФ в виде быстродействующих конденсаторов для каждых четырех сигналов, пересекающих плоскость. Конденсатор должен быть размещен на расстоянии, не больше чем 0,25 дюймов от точки пересечения [9].

Импеданс

Ненагруженный характеристический импеданс (Z_0) разделяемых сигнальных PCI-линий на плате расширения должен находиться в пределах 60—100 Ом, если входная емкость устройства C_{in} превышает 8 пФ. Если же значение емкости C_{in} равно или меньше 8 пФ, то диапазон допустимых значений Z_0 составляет 51—100 Ом [9, 21].

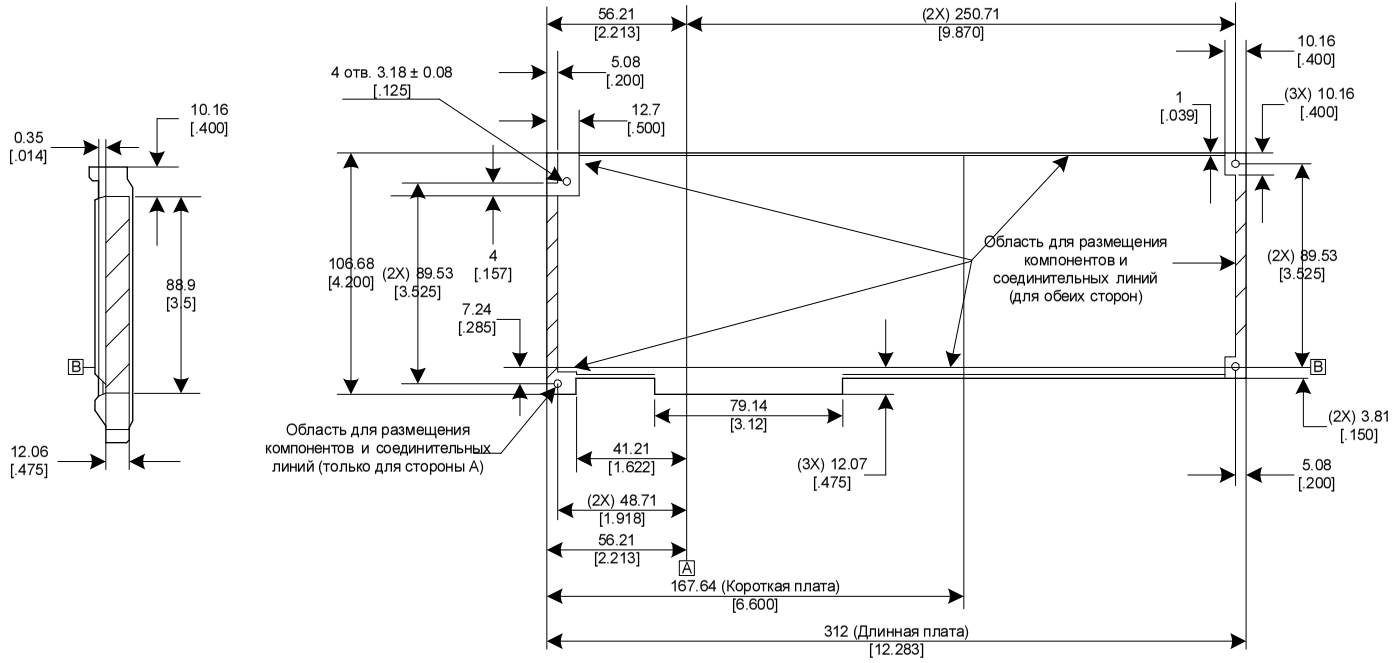
Нагрузка линии

Общие, или разделяемые, линии на плате расширения должны быть ограничены одной нагрузкой. Нарушение длины линии платы расширения или пределов нагрузки будет приводить к искажениям сигналов и ошибкам всей системы. Нарушениями спецификации для плат расширения являются:

- соединения ПЗУ-расширения (expansion ROM) непосредственно (или через приемопередатчики шины) с любыми выводами PCI;
- размещение двух или больше PCI-устройств на плате расширения без промежуточного элемента — PCI-to-PCI (также размещенного на плате расширения);
- размещение любой логики (в дополнение к единственному PCI-устройству), которое отслеживает состояние PCI-выводов;
- использование компонентов, которые соединяют с линией PCI больше чем одну нагрузку; например, компоненты с отдельными линиями адреса и данных;
- применение компонентов, имеющих емкость больше 10 пФ на каждый вывод;
- соединение нагрузочных резисторов или каких-либо дискретных устройств с линиями PCI, если они не размещаются после моста PCI-to-PCI. Группа линий SMBus освобождена от этого требования.

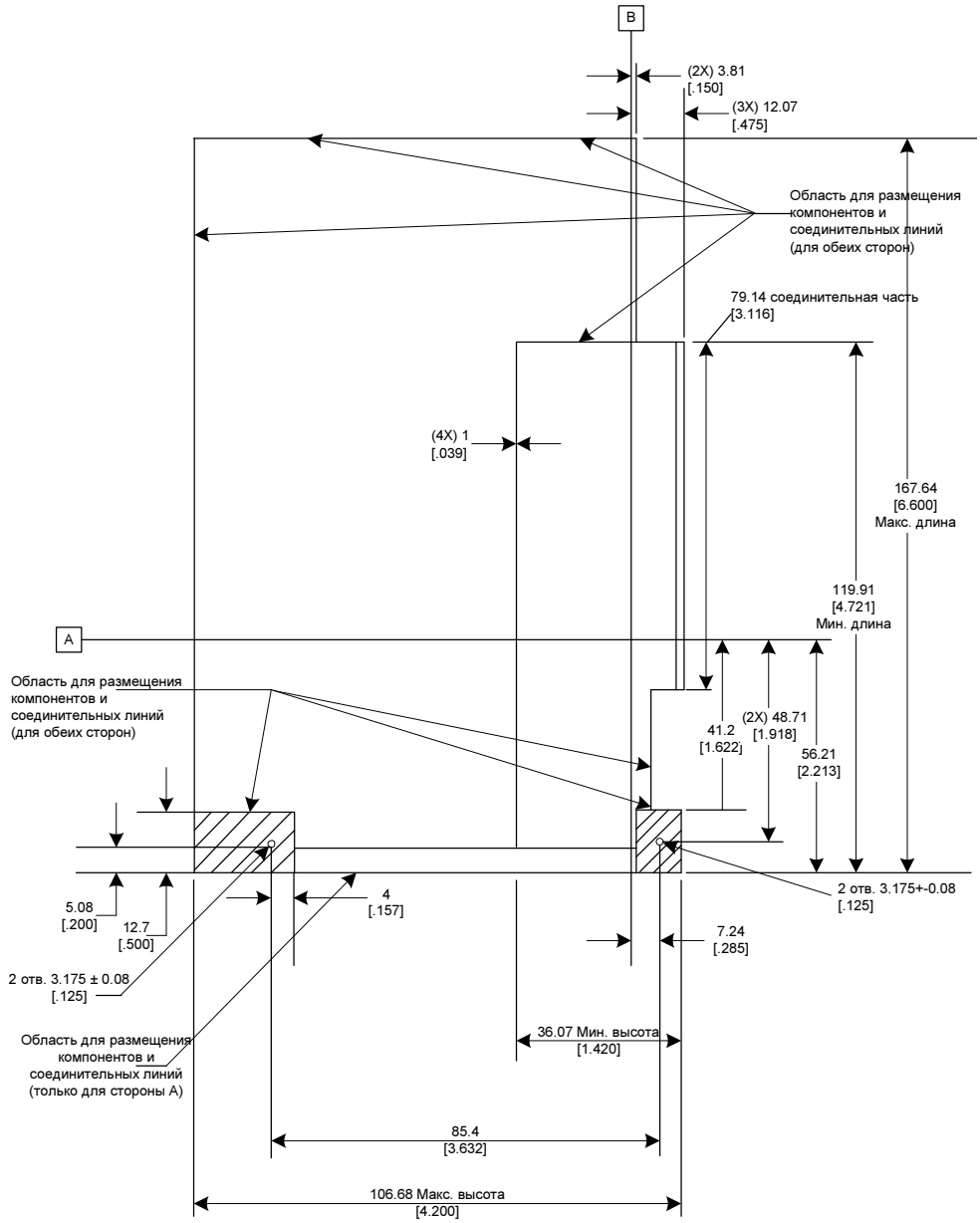
Конструктивные особенности

Размеры и физические параметры плат расширения показаны на рис. 10.1—10.4. Максимальная высота компонента на основной компонентной стороне PCI-платы расширения не должна превышать 0,570 дюймов (14,48 мм). Макси-



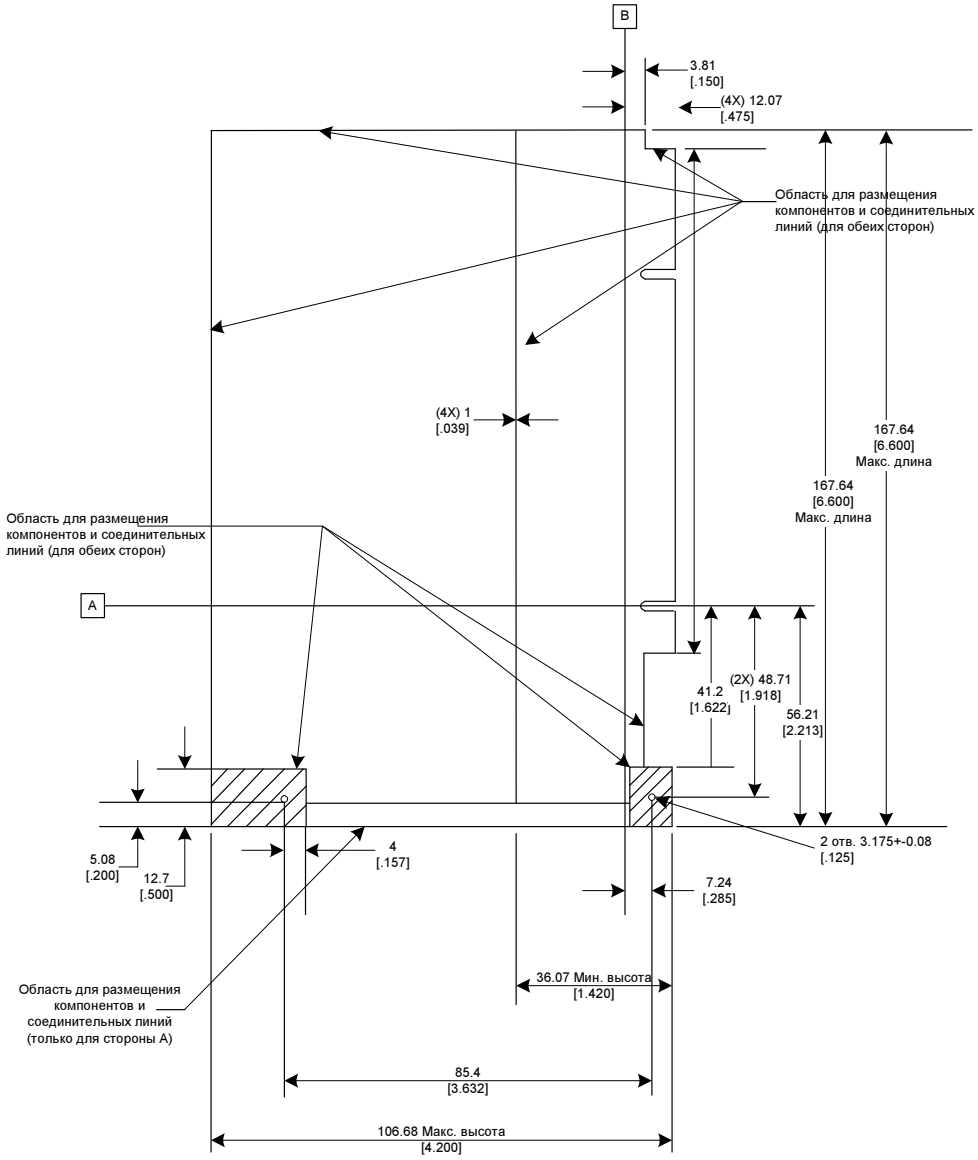
* Первое значение для всех размеров указано в миллиметрах, второе в дюймах

Рис. 10.1. 32-битная плата расширения для сигнальной среды 3,3 В



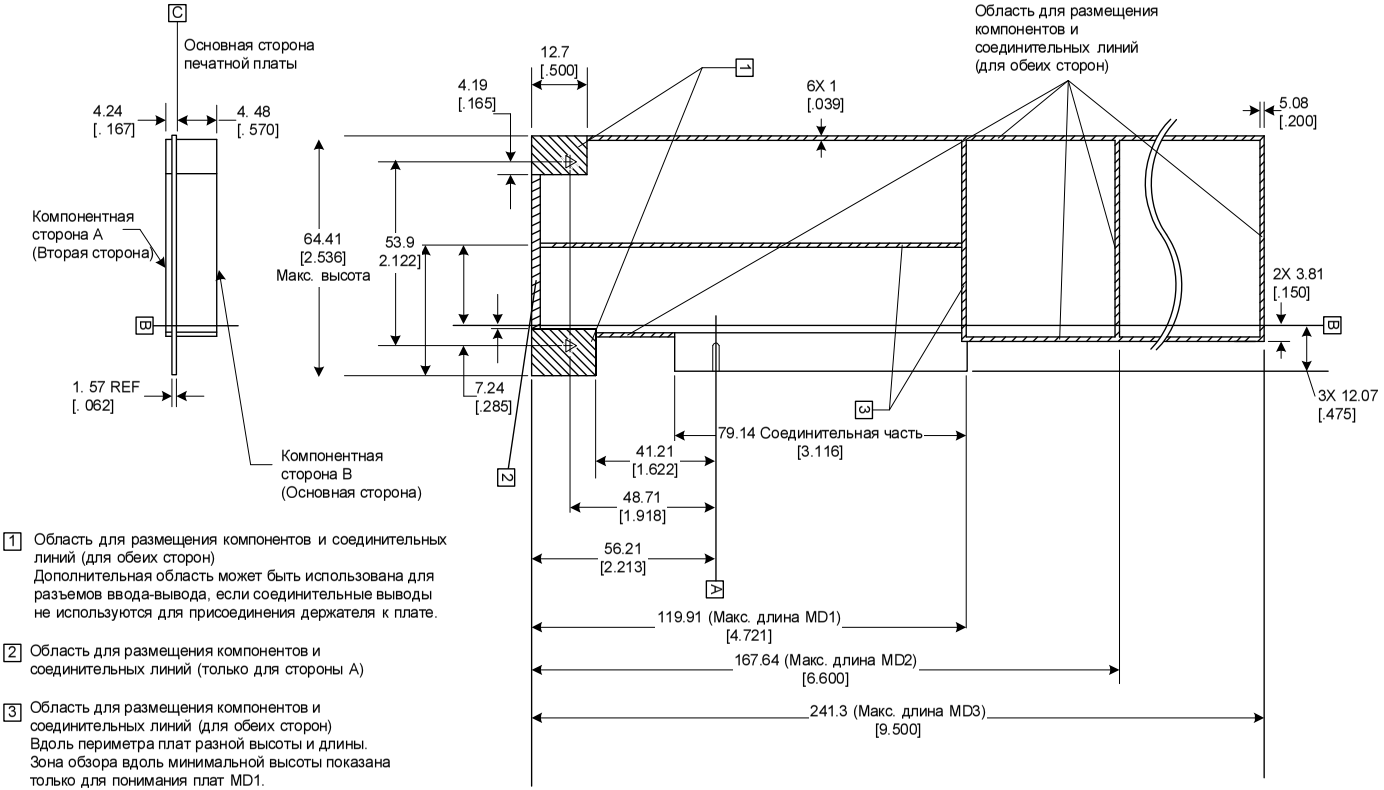
* Первое значение для всех размеров указано в миллиметрах, второе в дюймах

Рис. 10.2. 32-битная плата расширения (укороченная) для сигнальной среды 3,3 В



* Первое значение для всех размеров указано в миллиметрах, второе в дюймах

Рис. 10.3. 64-битная плата расширения (укороченная) для сигнальной среды 3,3 В



1 Область для размещения компонентов и соединительных линий (для обеих сторон)
Дополнительная область может быть использована для разъемов ввода-вывода, если соединительные выводы не используются для присоединения держателя к плате.

2 Область для размещения компонентов и соединительных линий (только для стороны A)

3 Область для размещения компонентов и соединительных линий (для обеих сторон)
Вдоль периметра плат разной высоты и длины. Зона обзора вдоль минимальной высоты показана только для понимания плат MD1.

Рекомендуется изготавливать платы с максимальными параметрами высоты для улучшения вентиляции, что позволит продлить срок службы платы расширения.

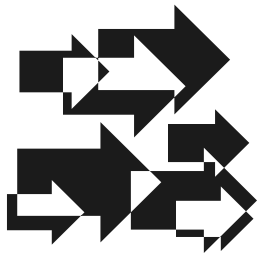
Допуск, не отмеченный специально, составляет 0.127 мм+-[0.0005 дюймов]

* Первое значение для всех размеров указано в миллиметрах, второе в дюймах

Рис. 10.4. 32-битная плата расширения для сигнальной среды 3,3 В (малый контур)

мальная высота компонента на обратной стороне платы расширения не должна превышать 0,105 дюймов (2,67 мм). Параметр "A" используется на иллюстрациях, чтобы определить размещение платы расширения относительно системной платы и интерфейса корпуса: задней части корпуса и направляющей для вставки платы. Линия "A" проходит через ключ на уровне платы расширения и через ключ на разъеме системной платы. На рис. 10.1—10.4 представлены размеры и допуски поддерживаемых типов плат расширения. Спецификация версии 2.3 поддерживает только платы расширения для сигнальной среды 3,3 В или универсальные платы расширения с ключом [9].

ГЛАВА 11



Конфигурирование и обслуживание устройств

Пространство конфигурации

Одним из требований, предъявляемым к вычислительным системам, является гибкость конфигурирования. Поскольку основными элементами шины являются внешние устройства, то должна быть обеспечена возможность конфигурирования устройств на шине. При этом желательно, чтобы устройство предоставляло минимальные функции управления: включение/выключение, изменение адресов ресурсов, линии прерывания и т. д. Кроме прочего, необходимо располагать информацией о состоянии устройства, а также обеспечить метод идентификации устройства вплоть до получения информации о версии выпуска и серийном номере. Все перечисленные требования реализованы на шине PCI в виде пространства конфигурации. Данное *пространство конфигурации* реализовано аппаратно и представляет собой набор регистров, содержащихся в каждом устройстве.

В данном разделе описана организация регистров пространства конфигурации, определена структура записей или иначе шаблон 256-байтного пространства. Пространство конфигурации разделено на стандартную предопределенную область заголовка и область, определяемую устройством. Организация области устройства определяется разработчиком. Устройства реализуют только необходимые регистры в каждой области. Пространство конфигурации устройства должно быть доступно в любой момент времени, а не только во время загрузки системы. Стандартная область заголовка состоит из полей, которые уникально идентифицируют устройство и обеспечивают общее управление устройством. Стандартный заголовок разделен на две части. Структура первых 16 байт одинакова для всех типов устройств. Остальные байты могут иметь различное расположение в зависимости от основной функции, которую

поддерживает устройство. Поле **Header Type**, расположенное по смещению 0Eh, определяет тип заголовка. В настоящее время определены три типа заголовка: "00h" , структура которого показана на рис. 11.1, тип "01h" для мостов PCI-to-PCI и тип "02h", который определен для мостов CardBus. Системное программное обеспечение может определять фактическое наличие устройств на шине путем сканирования, т. е. опроса всей шины. Для определения наличия устройства на шине конфигурационное ПО должно прочитать поле **Vendor ID** возможного устройства в каждом PCI-слоте. В свою очередь главная шина должна однозначно сообщить о попытках чтения поля **Vendor ID** несуществующих устройств. При попытке чтения регистров пространства конфигурации несуществующих устройств шина должна вернуть значение 0FFFFh. Значение 0FFFFh является недействительным значением поля **Vendor ID**, тем более что это равносильно установке всех единиц на шине данных. Такие операции чтения будут завершены с типом "Master-Abort". Все PCI-устройства должны рассматривать записи в зарезервированные регистры пространства конфигурации как пустые операции; т. е. доступ должен быть выполнен как обычно на шине, данные недействительны. Операции чтения из зарезервированных или нереализованных регистров должны быть выполнены стандартно, а именно — возвращено значение "0". На рис. 11.1 представлена структура стандартного заголовка типа "00h" 256-байтного пространства конфигурации. Устройства должны размещать любые необходимые специфические регистры в пространстве конфигурации после стандартного заголовка. Вся нумерация многобайтных полей производится в порядке "little endian"; т. е. младшие адреса содержат младшие части поля. Программное обеспечение должно это учитывать, чтобы правильно работать с полями, которые имеют биты, зарезервированные для будущего использования. При чтении программное обеспечение должно использовать соответствующие маски, чтобы извлечь определенные биты, и не должно применять зарезервированные биты, содержащие какое-либо значение. При записи программное обеспечение обязано гарантировать, что значения зарезервированных битов сохранены; т. е. значения зарезервированных позиций бита должны сначала быть прочитаны, объединены с новыми значениями для других позиций бита и затем записаны обратно [9].

Все PCI-совместимые устройства должны поддерживать поля **Vendor ID**, **Device ID**, **Command**, **Status**, **Revision ID**, **Class Code** и **Header Type** в стандартном заголовке. Реализация других регистров в стандартном заголовке типа "00h" является необязательной (т. е. они могут рассматриваться как зарезервированные) в зависимости от функциональных возможностей устройства. Если устройство поддерживает функцию, с которой связан регистр, то устройство должно реализовать его по определенному смещению.

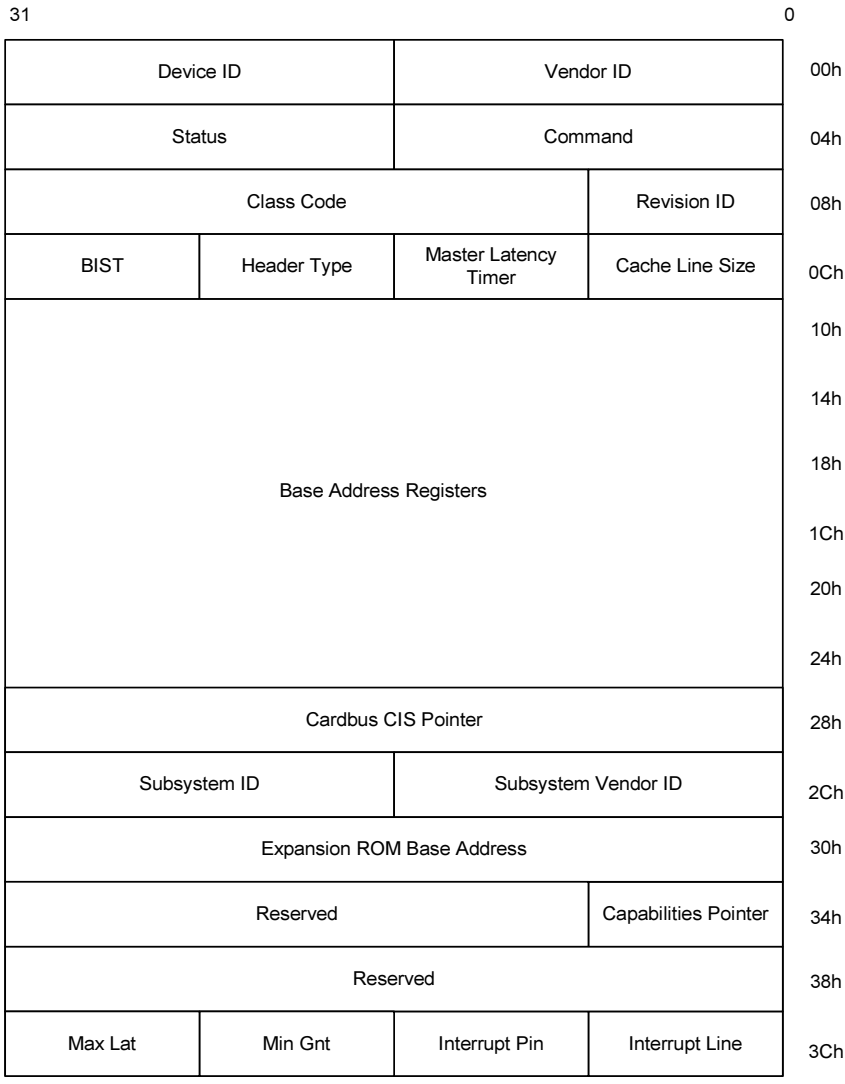


Рис. 11.1. Заголовок пространства конфигурации типа "00h"

Функции пространства конфигурации

Шина PCI предоставляет широкие возможности по конфигурированию системы. Для реализации этих возможностей все PCI-устройства должны обеспечить некоторые обязательные функциональные возможности, которые может использовать системное конфигурационное ПО. Данный раздел описыва-

ет функции, которые должны поддерживаться PCI-устройствами через регистры, определенные в стандартном заголовке. Точный формат этих регистров (т. е. число реализованных битов) определяется устройством. Тем не менее необходимо выполнение некоторых общих правил. Все регистры должны быть доступны для чтения, а возвращенные данные должны указать значение, фактически используемое устройством. Пространство конфигурации предназначено для реализации функций конфигурации, инициализации и функции обработки фатальной ошибки. Их использование ограничивается инициализационным ПО и ПО, предназначенным для обработки ошибок. Для управления регистрами устройства ПО должно использовать доступы к пространству ввода-вывода и/или пространству памяти [9].

Идентификация устройства

Для идентификации устройства предназначены пять полей в стандартном заголовке. Все PCI-устройства обязаны реализовать эти поля. Таким образом, универсальное конфигурационное ПО будет в состоянии определить и идентифицировать устройства, доступные на шине (ах) PCI системы. Данные регистры, определенные в стандартном заголовке, доступны только для чтения.

- **Vendor ID**. Поле идентифицирует изготовителя устройства. Действующие идентификаторы производителей выделяются комитетом PCI SIG, чтобы гарантировать уникальность. Значение 0FFFFh для данного поля недействительно, т. к. говорит об отсутствии устройства.
- **Device ID**. Поле идентифицирует конкретное устройство. Идентификатор выделяется производителем.
- **Revision ID**. Этот регистр определяет версию конкретного устройства. Значение выделяется производителем. Допустимо нулевое значение. Это поле должно рассматриваться как определяемое производителем дополнение к полю **Device ID**.
- **Header Type**. Данный байт идентифицирует расположение второй части стандартного заголовка (начинающейся с байта по смещению 10h в пространстве конфигурации), а также указывает, содержит ли устройство несколько функций. Бит 7 в этом регистре используется, чтобы идентифицировать многофункциональное устройство. Если значение бита равно "0", то устройство однофункциональное. Если же значение бита равно "1", то устройство многофункциональное. Биты с 6 до 0 идентифицируют структуру второй части стандартного заголовка. Значение "00h" определяет структуру, показанную на рис. 11.1. Значение "01h" определено для мостов PCI-to-PCI и описано в спецификации "PCI to PCI Bridge Architecture Specification". Значение "02h" определено для мостов CardBus и определено в спецификации "PC Card Standard". Все другие значения зарезервированы.

- **Class Code**. Регистр "код класса" доступен только для чтения и используется, чтобы идентифицировать основную функцию устройства и, в некоторых случаях, определенный программный интерфейс на уровне регистров. Регистр разбит на три поля каждое размером в один байт. Старший байт (по смещению 0Bh) — основной код класса, который классифицирует тип функции, реализованной устройством. Средний байт (по смещению 0Ah) — код подкласса, который идентифицирует более точно функцию устройства. Младший байт (по смещению 09h) идентифицирует определенный программный интерфейс на уровне регистров (если таковой имеется) таким образом, чтобы независимое от устройства программное обеспечение могло взаимодействовать с устройством. Значения для основного класса, подкласса и программного интерфейса описаны в *приложении 2*. Все остальные значения зарезервированы.

Управление устройством

Регистр Command обеспечивает "грубое" управление функцией генерирования и обработки циклов устройством PCI. Когда в этот регистр записано значение "0", устройство логически отсоединяется от шины PCI для всех доступов, кроме конфигурационных. Все устройства обязаны поддерживать данный основной уровень функциональных возможностей. Индивидуальные биты в регистре Command могут или не могут быть реализованы в зависимости от функциональных возможностей устройства. Например, устройства, которые не реализуют пространство ввода-вывода, не должны реализовывать перезаписываемый бит по смещению 0 регистра Command. Обычно при подаче питания все биты этого регистра содержат значения "0", но возможны некоторые исключения. Структура регистра показана на рис. 11.2, назначение различных битов в регистре Command описано далее.

- **IO Space** (бит 0). Управляет откликом устройства на доступы в пространство ввода-вывода. Значение "0" отключает устройство. Значение "1" разрешает устройству отвечать на доступы в ПВВ. Состояние после сброса (по умолчанию) — "0".
- **Memory Space** (бит 1). Управляет откликом устройства на доступы в пространство памяти. Значение "0" отключает устройство. Значение "1" разрешает устройству отвечать на доступы в пространство памяти. После сброса установлен в "0".
- **Bus Master** (бит 2). Управляет способностью устройства действовать как мастер на шине PCI. Значение "0" запрещает устройству генерировать PCI доступы. Значение "1" разрешает устройству функционировать в качестве мастера шины. После сброса установлен в "0".
- **Special Cycles** (бит 3). Управляет действиями устройства во время операций специального цикла. Значение "0" заставляет устройство игнориро-

вать все операции специального цикла. Значение "1" разрешает устройству отслеживать операции специального цикла. После сброса установлен в "0".

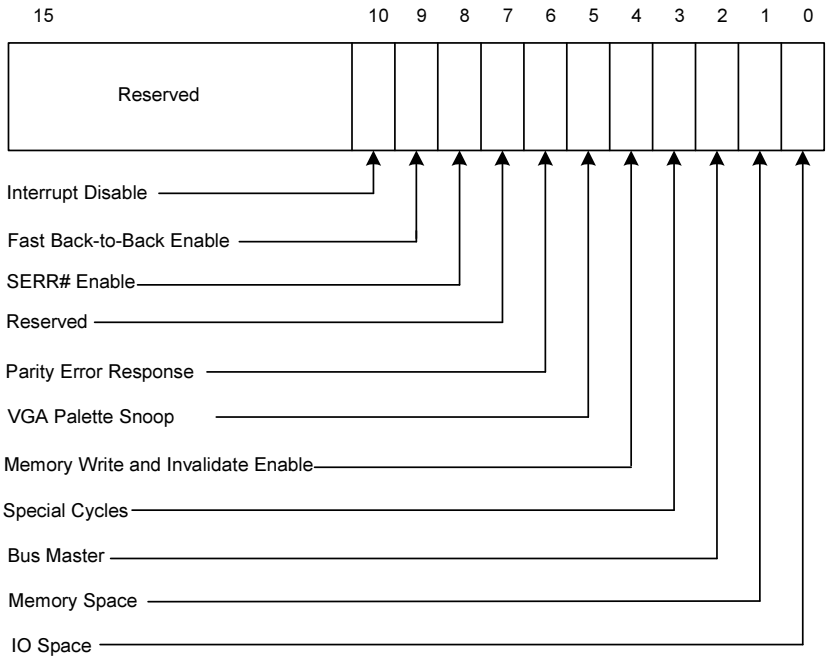


Рис. 11.2. Структура регистра Command

- **Memory Write and Invalidate Enable** (бит 4). Бит разрешения для использования команды "Memory Write and Invalidate". Когда данный бит установлен в "1", мастера могут генерировать эту команду. Если значение данного бита "0", то вместо нее должна использоваться команда "Memory Write". После сброса установлен в "1". Этот бит должен быть реализован всеми мастерами, которые могут генерировать команду "Memory Write and Invalidate".
- **VGA Palette Snoop** (бит 5). Данный бит отвечает за обработку VGA совместимыми и другими графическими устройствами доступов в регистры палитры VGA. Значение данного бита "1" разрешает отслеживание палитры (т. е. устройство не реагирует на записи в регистры палитры и отслеживает данные). При установке данного бита в "0" устройства должны рассматривать доступы записи палитры как все остальные доступы. VGA совместимые устройства должны реализовать данный бит.

- **Parity Error Response** (бит 6). Данный бит управляет откликом устройства на ошибки четности. Если данный бит установлен, то устройство должно обрабатывать ошибки четности. Если значение данного бита "0", то при обнаружении ошибки четности устройство устанавливает бит состояния "Detected Parity Error" (бит 15 регистра статуса устройства), но не выставляет сигнал PERR# и продолжает функционировать в обычном режиме. После сброса установлен в "0". Устройства, проверяющие четность, должны реализовать этот бит. Устройства должны генерировать четность, даже если отключена проверка четности.
- **Reserved** (бит 7). Этот бит зарезервирован и должен всегда содержать значение "0"[1].
- **SERR# Enable** (бит 8). Бит разрешения для драйвера линии SERR#. Значение "0" отключает драйвер SERR#, значение "1" включает. После сброса установлен в "0". Должен быть реализован всеми устройствами, которые имеют вывод SERR#. Ошибки четности адреса сообщаются, только если этот бит, а также бит 6 установлены в "1".
- **Fast Back-to-Back Enable** (бит 9). Бит управляет способностью мастера выполнять быстрые транзакции типа back-to-back к различным устройствам. Реализация данного бита необязательна, бит доступен для чтения/записи. Инициализационное ПО будет устанавливать этот бит, если все цели поддерживают быстрые транзакции back-to-back. Значение "1" означает, что мастеру разрешено генерировать быстрые транзакции типа back-to-back к различным агентам. Значение "0" означает, что такие транзакции разрешено генерировать только к определенным агентам. После сброса бит установлен в "0".
- **Interrupt Disable** (бит 10). Данный бит запрещает устройствам или функциям выставлять сигнал INTx#. Значение "0" для данного бита разрешает установку сигнала INTx#, значение "0"— запрещает. После сброса бит установлен в "0".
- **Reserved** (биты 11—15). Эти биты зарезервированы.

Статус устройства

Регистр Status предназначен для записи информации о событиях, связанных с шиной PCI. В зависимости от функциональных возможностей, устройства могут не реализовывать все биты регистра. Например, устройство, которое

[1] Бит 7 не может быть использован устройствами. Этот бит определен в устаревшей версии спецификации, устройства могли реализовывать его как постоянное значение "0", "1" или как бит для чтения/записи.

функционирует как цель, но никогда не будет использовать тип завершения типа "Target-Abort, не должно реализовывать бит 11. Зарезервированные биты должны быть доступны только для чтения и при чтении возвращать ноль. Чтение из этого регистра выполняется как обычно. Запись несколько отличается тем, что биты могут быть сброшены, но не установлены. Запись значения "1" по соответствующему смещению бита означает его сброс. Например, чтобы сбросить бит 14 и не изменить остальные биты, в регистр должно быть записано значение 0100000000000000b. Структура регистра Status приведена на рис. 11.3, а далее описано назначение его различных битов [9].

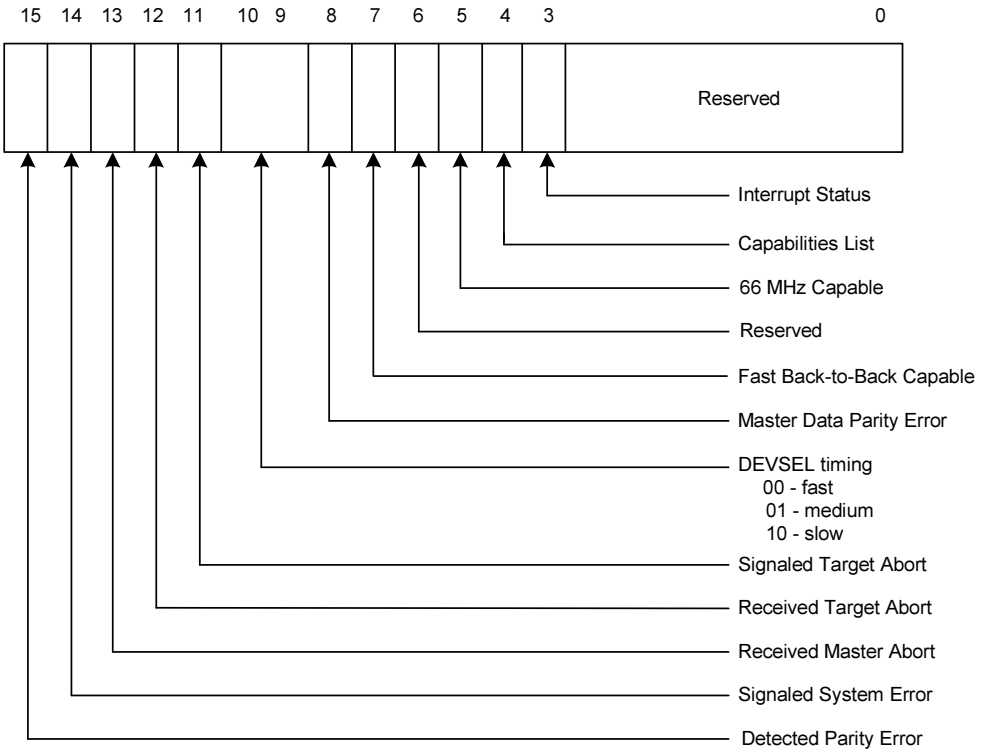


Рис. 11.3. Структура регистра Status

- **Reserved** (биты 0—2). Эти биты зарезервированы.
- **Interrupt Status** (бит 3). Бит состояния прерывания устройства или функции, доступен только для чтения. Сигнал INTx# устройства или функции может быть выставлен, только если значение бита "Interrupt Disable" регистра команд равно "0" и значение бита "Interrupt Status" равно "1". Установка бита "Interrupt Disable" в "1" не влияет на состояние данного бита.

- **Capabilities List** (бит 4). Реализация указателя на связанный список новых возможностей по смещению 34h. Данный бит доступен только для чтения и его реализация необязательна. Значение "0" указывает, что список не доступен. Значение "1" указывает, что величина, прочитанная по смещению 34h, является указателем в конфигурационное пространство на связанный список новых возможностей.
- **66 MHz Capable** (бит 5). Способность устройства работать на частоте 66 МГц. Данный бит доступен только для чтения и его реализация необязательна. Значение "0" указывает, что устройство поддерживает частоту 33 МГц, значение "1" указывает на способность устройства работать на частоте 66 МГц.
- **Reserved** (бит 6). Данный бит зарезервирован[2].
- **Fast Back-to-Back Capable** (бит 7). Бит указывает, может или нет цель принимать быстрые транзакции типа back-to-back, когда транзакции выполняются не для конкретного агента. Данный бит доступен только для чтения и его реализация необязательна. Бит должен быть установлен в "1", если устройство может принять эти транзакциии, и в "0" — в противном случае.
- **Master Data Parity Error** (бит 8). Данный бит реализуется только мастерами шины. Он устанавливается при следующих условиях:
  - агент шины выставил свой сигнал PERR# (при чтении) или обнаружил установку сигнала PERR# (на записи);
  - агент установил бит, действуя как мастер шины для операции, в которой произошла ошибка;
  - установлен бит "Parity Error Response" (регистр Command).
- **DEVSEL timing** (биты 9—10). Данные биты указывают время установки сигнала DEVSEL#. Правила выбора устройства определяют три возможных значения для установки сигнала DEVSEL#. Значение данных битов "00b" соответствует быстрой установке сигнала — "fast", значение "01b" соответствует средней — "medium", а значение "10b" медленной — "slow" (значение "11b" зарезервировано). Данные биты доступны только для чтения и должны указывать наименьший интервал времени, через который устройство выставит сигнал DEVSEL# для любых команд шины, кроме *Configuration Read* и *Configuration Write*.
- **Signaled Target Abort** (бит 11). Данный бит должен быть всегда установлен целью при завершении транзакции с типом "Target-Abort". Устройст-

[2] В версии 2.1 спецификации данный бит указывал на поддержку устройством функций, определяемых пользователем.

ва, которые никогда не будут использовать этот тип завершения, могут не реализовывать данный бит.

- **Received Target Abort** (бит 12). Данный бит должен быть всегда установлен мастером, когда его транзакция завершена с типом "Target-Abort". Все устройства, которые функционируют как мастер, должны реализовать данный бит.
- **Received Master Abort** (бит 13). Данный бит должен быть всегда установлен мастером, когда его транзакция (кроме специального цикла) завершается с типом "Master-Abort". Все устройства, которые функционируют как мастер, должны реализовать данный бит.
- **Signaled System Error** (бит 14). Этот бит должен быть всегда установлен устройством при установке сигнала SERR#. Устройства, которые никогда не выставляют сигнал SERR#, могут не реализовывать этот бит.
- **Detected Parity Error** (бит 15). Этот бит должен быть всегда установлен устройством при обнаружении ошибки четности, даже если обработка ошибок четности отключена (посредством бита 6 регистра команд Command).

Смешанные регистры

В этом разделе описаны регистры, которые являются независимыми от устройства и должны быть реализованы только теми устройствами, которые обеспечивают указанную функцию.

CacheLine Size

Данный регистр определяет размер системной строки кэш-памяти, размер указан в двойных словах. Этот регистр доступен для чтения/записи и должен быть реализован мастер-устройствами, которые поддерживают команду *Memory Write and Invalidate*. Значение в этом регистре также используется мастер-устройствами, чтобы решить, когда использовать команды *Read*, *Read Line* или *Read Multiple* для обращения в память. Цели, которые допускают блочную передачу в операциях с памятью в режиме обратной строки кэша, должны реализовать этот регистр. За счет него цели будет известно, когда последовательность блока возвращается к началу строки кэша. После сброса значение данного регистра должно быть "0". Устройство может ограничить диапазон поддерживаемых размеров строки кэша. Например, оно может разрешить поддержку от 2 до 128 строк. Если в регистр CacheLine Size записано неподдерживаемое значение, то устройство должно вести себя, как при записи нуля.

Latency Timer

Этот регистр определяет, в единицах тактов шины PCI, значение таймера задержки для данного мастера шины. Регистр доступен для записи и должен быть реализован любым мастером, который может передавать блоки больше двух фаз данных. Регистр может быть реализован в качестве доступного только для чтения для устройств, которые передают блоки размером в две или меньше фазы данных, но значение должно быть аппаратно ограничено 16 (или меньше). В типичной реализации используются пять старших битов (оставляя три младших, как доступные только для чтения), что приводит к частоте дискретизации таймера в восемь тактов. После сброса регистр должен быть установлен в "0".

Built-in Self Test (BIST)

Данный регистр используется для управления и получения информации о состоянии внутреннего теста устройства — *BIST*. Реализация регистра необязательна. Устройства, которые не поддерживают тест BIST, должны всегда возвращать значение 0 (т. е. рассматривать его как зарезервированный регистр). Устройство, BIST которого вызван, не должно влиять на нормальное действие шины PCI. Структура регистра приведена на рис. 11.4, а назначение различных битов в регистре BIST описано далее.

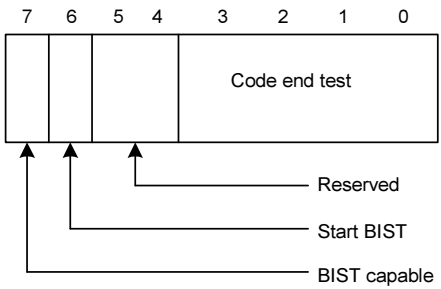


Рис. 11.4. Структура регистра BIST

- **BIST capable** (бит 7). Бит возвращает "1", если устройство поддерживает тест BIST, и "0", если не поддерживает.
- **Start BIST** (бит 6). Для активизации BIST данный бит должен быть установлен в "1". Устройство сбрасывает его после выполнения BIST. ПО может вывести из строя устройство, если BIST не выполнен по истечении 2 секунд.
- **Reserved** (биты 5—4). Зарезервированы. Устройство возвращает "0".

- **Code end test** (биты 3—0). Значение "0" означает, что устройство успешно завершило тестирование. Величина, отличная от "0", означает, что устройство неисправно. Конкретное значение кода указывает на характер или причину неисправности.

CardBus CIS Pointer

Данный регистр используется теми устройствами, которые хотят использовать микросхему на обоих шинах — CardBus и PCI. Реализация регистра необязательна. Поле используется, чтобы указать на структуру "Card Information Structure" (*CIS*) для платы CardBus. Детальное описание структуры CIS приведено в спецификации "PCMCIA v2.10 Specification" в главе "Card Metaformat" и описывает типы предоставляемой информации и ее организацию.

Interrupt Line

Регистр Interrupt Line 8-битный регистр, используемый для сообщения информации о линии прерывания. Регистр доступен для чтения/записи и должен быть реализован любым устройством (или функцией), которое использует вывод прерывания INTx#. Процедура POST (Power-On Self Test) запишет информацию о линии прерывания в этот регистр, когда оно инициализирует и конфигурирует систему. Значение в этом регистре говорит, какой вход системного контроллера(ов) прерываний связан с выводом прерывания устройства. Само устройство не использует это значение, оно в основном предназначено для драйверов устройств и операционных систем. Драйверы устройства и операционные системы могут использовать эту информацию, чтобы определить приоритет и вектор прерывания. Значение в этом регистре зависит от особенностей системной архитектуры[3].

Interrupt Pin

Регистр "Interrupt Pin" содержит информацию о том, какие выводы прерываний использует устройство (или функция). Значение "1" соответствует выводу INTA#, значение "2" — выводу INTB#, значение "3" — INTC#, а значение "4" — INTD#. Устройства (или функции), которые не используют выводы прерывания, должны поместить в данный регистр значение "0". Значения от "05h" до "FFh" зарезервированы. Регистр доступен только для чтения.

---

[3] Для систем *x*86 значение в этом регистре соответствует номеру IRQ (0—15) стандартной двойной конфигурации контроллера прерываний 8259. Значение "255" определено как означающее "неизвестное" или "не присоединенное" к контроллеру прерываний. Значения от 15 до 254 зарезервированы.

MIN_GNT и MAX_LAT

Данные регистры используются для определения значений, которые устройство предполагает установить для таймера времени ожидания. Оба регистра имеют размер 1 байт и доступны только для чтения. Для обоих регистров содержащееся в них значение определяет период времени в единицах ¼ микросекунды. Значения "0" указывают, что устройство не имеет основных требований для таймеров задержки.

Регистр MIN_GNT используется для того, чтобы определить величину периода блока для устройства, учитывая, что частота шины равна 33 МГц. Регистр MAX_LAT используется для того, чтобы определить, как часто устройство должно получать доступ к шине PCI. Устройствам следует определить значения, которые позволят им наиболее эффективно использовать шину PCI и собственные внутренние ресурсы. Значения должны выбираться из предположения, что цель не вставляет состояний ожидания.

Пример выбора регистров MIN_GNT и MAX_LAT

В качестве примера рассмотрим контроллер Fast Ethernet (100 Мбит/с), который имеет 64-байтный буфер для каждого направления передачи. Оптимальное использование этих внутренних ресурсов достигается, когда устройство рассматривает каждый буфер как два 32-байтных *буфера типа "ping-pong"*, т. е. как два буфера с попеременным переключением. Каждый 32-байтный буфер содержит данные размером восемь двойных слов для передачи, таким образом, потребуется восемь фаз данных на шине PCI. Эти восемь фаз данных переводятся в ¼-микросекунды для частоты 33 МГц ($8 \times 1/33 \times 10^6$ = 0,2424...10^{-6} с = 0,2424... мкс), так что значение регистра MIN_GNT для этого устройства будет равно "1". При перемещении данных устройство должно будет очищать или заполнять 32-байтный буфер каждые 3,2 мкс (принимая во внимание производительность 10 Мбайт/с). Это соответствовало бы значению "12" в регистре MAX_LAT.

Subsystem Vendor ID и Subsystem ID

Данные регистры используются, чтобы однозначно идентифицировать плату расширения или подсистему, где находится PCI-устройство. Они предоставляют механизм для производителей плат расширения, позволяющий отличить платы расширения друг от друга даже в том случае, если платы расширения имеют одинаковые PCI-контроллеры и, следовательно, одинаковые значения регистровVendor ID и Device ID.

Данные регистры должны быть реализованы всеми PCI-устройствами за исключением тех, которые имеют основной класс 6 и подкласс 0—4 (0, 1, 2, 3, 4), или основной класс 8 и подкласс 0—3 (0, 1, 2, 3). Значение для поля Subsystem Vendor ID может быть получено от комитета PCI SIG и использу-

ется, чтобы идентифицировать производителя платы расширения или подсистемы (чипа)[4]. Значения для поля **Subsystem ID** определяются производителем.

Значения в эти регистры должны быть записаны до выполнения BIOS или любого программного обеспечения системы, обращающегося в пространство конфигурации PCI. Способ загрузки этих регистров не определен, но это может быть сделано во время производственного процесса, либо значения могут загружаться из внешней логики (например, serial ROMs и т. д.). Эти значения не должны загружаться программным обеспечением из дополнительной Flash-памяти (expansion ROM), потому что такое программное обеспечение не будет работать во время выполнения процедуры POST, а будет работать после. Устройства должны гарантировать, что данные действительны, прежде чем позволить выполнить чтение из этих регистров. Это может быть реализовано путем ответа "Retry" при любых доступах, пока данные не будут готовы. Если устройство предназначено для использования только на системной плате, то производитель системы может использовать специфическое системное ПО для инициализации этих регистров после каждого включения питания.

Capabilities Pointer

Этот регистр используется, чтобы указать на связанный список новых возможностей, реализованных этим устройством. Реализация данного регистра необязательна. Этот регистр действителен, только если установлен бит **Capabilities List** в регистре Status. При его реализации два младших бита зарезервированы и должны содержать значение "00b". Программному обеспечению надлежит маскировать эти биты перед использованием регистра в качестве указателя на первый вход в связанный список новых возможностей в пространстве конфигурации.

Базовые адреса

Наиболее важной функцией с точки зрения конфигурирования и простоты использования является способность перемещать PCI-устройства в адресных пространствах. При включении питания аппаратно-независимое программное обеспечение должно быть в состоянии определить, какие устройства присутствуют, построить согласованную карту адресов, и определить наличие до-

[4] Разработчикам требуется только одно поле **Vendor ID**. Это значение может быть использовано в обоих полях **Vendor ID** (offset 00h) или **Subsystem Vendor ID** (offset 2Ch) конфигурационного пространства. Если компания является производителем микросхемы, то используется поле **Vendor ID** (offset 00h), если компания является производителем плат расширения, то используется поле **Subsystem Vendor ID** (offset 2Ch). Если компания производит и то и другое, то в обоих полях будут одинаковые значения.

полнительных ПЗУ (ROM). Каждая из этих функций рассмотрена в следующих разделах [9].

Карта адресного пространства

Программное обеспечение, функционирующее после включения питания (BIOS, процедура POST), обязано строить карту адресов перед загрузкой операционной системы. Это означает, что оно должно определить, сколько памяти присутствует в системе и сколько адресного пространства требуется для контроллеров ввода-вывода в системе. После определения этой информации программное обеспечение может отобразить контроллеры ввода-вывода по корректным адресам и продолжить загрузку системы. Для того чтобы сделать отображение аппаратно-независимым способом, базовые регистры для отображения помещаются в стандартную часть заголовка пространства конфигурации. Бит 0 во всех регистрах Base Address доступен только для чтения и используется, чтобы определить, куда производится отображение — в память или пространство ввода-вывода. Значение "0" бита 0 соответствует отображению регистров базового адреса в пространство памяти, значение "1" — в пространство ввода-вывода. Структура регистра базового адреса для памяти приведена на рис. 11.5, структура регистра базового адреса для пространства ввода-вывода — на рис. 11.6. Назначение различных битов обоих регистров описано далее.

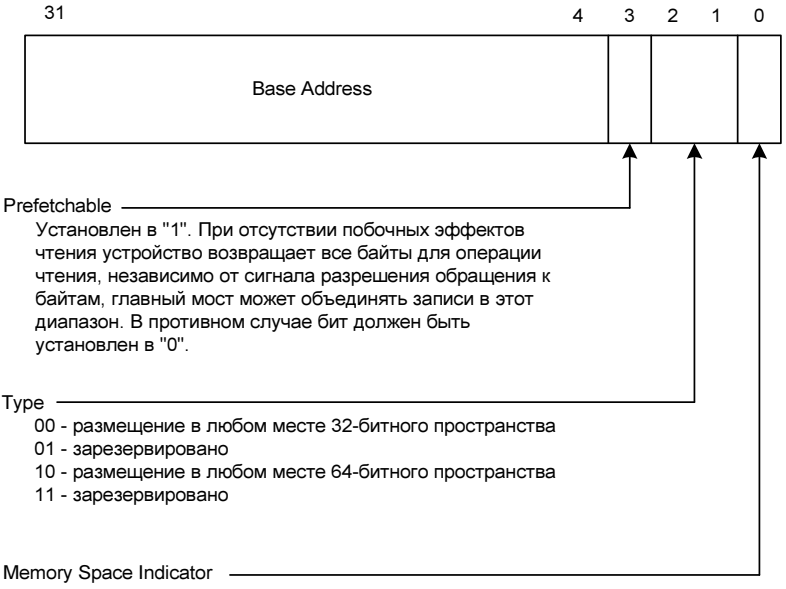


Рис. 11.5. Структура регистра базового адреса для пространства памяти

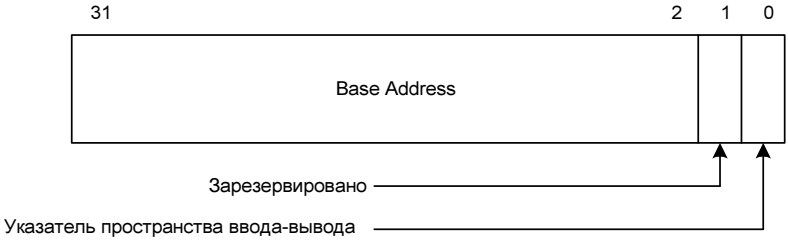


Рис. 11.6. Структура регистра базового адреса для пространства ввода-вывода

Регистры базового адреса, которые отображаются в пространство ввода-вывода, всегда имеют размер 32 бита, при этом бит 0 аппаратно установлен в "1". Бит 1 является зарезервированным и при чтении должен возвращать значение "0", остальные биты используются, чтобы отобразить устройство в пространство ввода-вывода. Регистры базового адреса, которые отображаются в пространство памяти, могут быть 32- или 64-битными (для обеспечения поддержки отображения в 64-битное адресное пространство) с нулевым значением бита 0. Для регистров базового адреса памяти биты 2 и 1 несут значимую информацию, их значения приведены в табл. 11.1. Бит 3 должен быть установлен в "1", если допускается предвыборка данных и сброшен в "0" в противном случае. Устройство может пометить диапазон как доступный для предвыборки, если: при чтении отсутствуют побочные эффекты, устройство возвращает все байты при чтении независимо от сигнала разрешения обращения к байтам, и главные мосты могут производить слияние байт при записи процессора в этот диапазон[5] без ошибок. Биты 0—3 доступны только для чтения [9].

Регистры базового адреса содержат два важных информационных поля: **Base Address** — базовый адрес, куда будет отображено устройство, и **Type** — тип, указывающий размер адресного пространства, требуемого устройству. Реализация двух полей в одном регистре достигается просто. Число реализованных устройством старших бит зависит от того, какое количество адресного пространства необходимо устройству. В реализованных старших битах хранится базовый адрес, чем достигается естественное выравнивание. Таким образом, значение базового адреса изменяется на величину требуемого диапазона, и при любых значениях базовых регистров диапазоны не пересекутся. 32-битный регистр обеспечивает один размер памяти, который является степенью 2

[5] Если устройство содержит адресное пространство, функционирующее как нормальная память, этот диапазон будет помечен как доступный для упреждающего чтения. Линейный кадровый буфер в графических устройствах является примером диапазона, который будет помечен как доступный для упреждающего чтения.

Таблица 11.1. *Значения битов 2—1 регистра базового адреса памяти*

Биты 2—1	Назначение
00	Размер регистра базового адреса 32 бита и отображение может быть произведено в любое место 4-гигабайтного пространства памяти
01	Зарезервировано[6]
10	Размер регистра базового адреса 64 бита и отображение может быть произведено в любое место 64-битного адресного пространства
11	Зарезервировано

и его значение может быть от 16 байтов до 2 Гбайт. Устройство, которому требуется 1 Мбайт адресного пространства (используя 32-битный регистр базового адреса) реализует старшие 12 бит регистра адреса, аппаратно выставляя "0" на других битах. Инициализационное программное обеспечение может определить, какого количества адресного пространства требует устройство, записывая значение "1" во все биты регистра и затем читая значение обратно. Устройство возвратит значение "0" во всех нереализованных битах. Нереализованные регистры базового адреса должны быть аппаратно сведены к нулю. Для отображения регистров устройства могут потреблять больше адресного пространства, чем им требуется. При этом дешифрация осуществляется вниз к 4-килобайтному пространству памяти, предполагаемой для устройств, которым требуется меньшее, чем это количество. Например, устройство, имеющее 64 байта регистров для отображения в пространство памяти, может потреблять до 4 Кбайт адресного пространства, чтобы минимизировать число бит в адресном дешифраторе. Устройства, которые действительно потребляют больше адресного пространства, чем они используют, не обязаны отвечать на неиспользованную часть этого адресного пространства. Устройства, которые отображают управляющие функции в пространство ввода-вывода, должны потреблять не больше, чем 256 байтов на каждый регистр базового адреса Base Address. Старшие 16 битов регистра базового адреса ввода-вывода могут быть сведены к нулю для устройств, предназначенных для 16-битных PC-совместимых систем. В любом случае производится полная 32-битная дешифрация адреса ввода-вывода [9].

Выбор размера регистра базового адреса

Рассмотрим пример определения размера адресного пространства. В первую очередь отключается дешифрация (ввода-вывода или памяти) через регистр

[6] В предыдущих версиях спецификации данное значение использовалось для поддержки пространства памяти ниже 1 Мбайта. Системное ПО должно правильно обрабатывать данное значение.

команд — Command. Затем программное обеспечение сохраняет значение, содержащееся в регистре базового адреса Base Address, записывает значение "0FFFFFFFh" в регистр и читает из регистра. Вычисление размера из полученного 32-битного значения производится следующим образом. Сначала очищаются информационные биты (бит 0 для ввода-вывода, биты 0—3 для памяти), затем все 32 бита инвертируются (логическое НЕ), а затем все значение увеличивается на 1. Полученное 32-битное значение и есть размер диапазона памяти или ввода-вывода. Старшие 16 битов результата игнорируются, если регистр базового адреса предназначен для отображения в пространство ввода-вывода и биты 16—31 возвращают ноль при чтении. После определения размера адресного пространства в регистре базового адреса восстанавливается первоначальное значение и только потом включается дешифрация в регистре команд устройства.

Регистры базового адреса размером 64 бита (для отображения в память) обрабатываются так же, за исключением того, что второй регистр на 32 бита рассматривают как расширение первого; т. е. биты 32—63. Программное обеспечение записывает значение "0FFFFFFFFh" в оба регистра, читает из регистров, и объединяет результат в значение на 64 бита. Остальные действия выполняются уже с 64-битным значением.

Стандартный заголовок типа "00h" содержит шесть 32-битных регистров, выделенных для регистров базового адреса и начинающихся по смещению 10h в пространстве конфигурации. Устройство может использовать любой из них для реализации регистров базового адреса. 64-битный регистр базового адреса занимает два последовательных 32-битных регистра. Программное обеспечение, которое определяет наличие реализованных регистров базового адреса, должно искать их в диапазоне от 10h и до 24h. В большинстве случаев устройство будет требовать один диапазон памяти для его управляющих функций. Некоторые графические устройства могут использовать два диапазона: один для функций управления и другой для кадрового буфера. Устройство, которое хочет одновременно отобразить функции управления и в память и в пространство ввода-вывода, должно реализовать два регистра базового адреса (один для памяти и один для системы ввода-вывода). Драйвер такого устройства мог бы использовать только одно пространство, при этом другое пространство было бы не использовано. Рекомендуется всегда отображать функции управления устройством в пространство памяти [9].

Регистр базового адреса ПЗУ расширения

Некоторые PCI-устройства, особенно те, которые предназначены для использования на платах расширения в архитектуре PC, нуждаются в микросхемах памяти типа EEPROM (и/или Flash) для хранения конфигурационной информации или ПО. Для хранения информации базового адреса и размера допол-

нительной постоянной памяти ROM в стандартном заголовке типа "00h" по смещению 30h определен четырехбайтный регистр базового адреса ПЗУ расширения — "Expansion ROM Base Address". Структура этого регистра представлена на рис. 11.7. Регистр функционирует подобно 32-битному регистру базового адреса за исключением младших его битов. Старшие 21 бита соответствуют старшим 21 битам базового адреса дополнительного ПЗУ (expansion ROM). Число битов (из этих 21), которое фактически реализует устройство, зависит от того, какое количество адресного пространства требуется устройству. Например, устройство, которому для отображения части ее дополнительного ПЗУ требуется область размером 64 Кбайт, реализовало бы в регистре старшие 16 битов, сведя младшие 5 (из этих 21) к нулю. Устройства, которые поддерживают дополнительное ПЗУ, должны реализовать этот регистр. Алгоритм определения размера адресного пространства, необходимого устройству, такой же, как и для регистров базовых адресов. Программное обеспечение записывает "1" во все биты регистра и затем читает из регистра. Все нереализованные биты вернут "0", эффективно определяя размер и требования выравнивания. Количество запрашиваемого устройством адресного пространства не должно превышать 16 Мбайт.

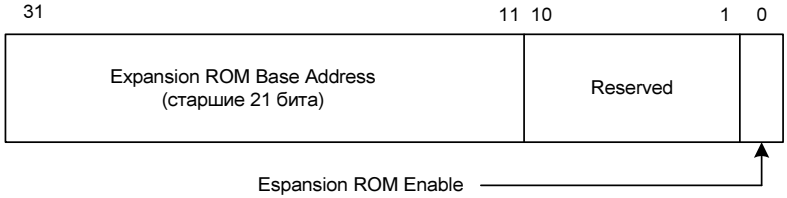


Рис. 11.7. Структура регистра базового адреса ПЗУ расширения

Бит 0 в регистре используется для управления доступом к дополнительному ПЗУ (expansion ROM) устройства. Значение "0" данного бита означает, что адресное пространство дополнительного ПЗУ устройства выключено. Значение "1" включает дешифрацию адреса, используя параметры в другой части основного регистра. Это позволяет использовать устройства с наличием или без дополнительного ПЗУ в зависимости от конфигурации системы. Бит пространства памяти в регистре команд имеет более высокий приоритет, чем бит разрешения дополнительного ПЗУ (бит 0). Устройство должно отвечать при доступе в его дополнительное ПЗУ, только если оба бита: бит пространства памяти (бит **Memory Space** регистра Command) и бит разрешения базового адреса (бит **Expansion ROM Enable** регистра базового адреса ПЗУ расширения) установлены в "1". После сброса бит устанавливается в "0". Чтобы минимизировать количество необходимых адресных дешифраторов, устройство может использовать один и тот же дешифратор для регистра базового адреса

дополнительного ПЗУ и других регистров базового адреса[7]. Когда включена дешифрация дополнительного ПЗУ, дешифратор используется для доступа к нему и аппаратно-независимое программное обеспечение не должно получить доступ к устройству через обычные регистры базового адреса.

"Жизненные" данные изделия

Vital Product Data (VPD) — информация, которая уникально определяет изделия типа аппаратных средств, программного обеспечения, и элементов микрокода системы. VPD обеспечивает систему информацией относительно различных модулей *FRUs* (Field Replaceable Unit — модулей перемещаемых полей), включая номер изделия, серийный номер и другую детальную информацию. VPD также обеспечивает механизм для хранения информации, такой, как данные производительности и сбоев относительно отслеживаемого устройства. Цель, с точки зрения системы, состоит в том, чтобы собрать эту информацию, читая ее из аппаратного обеспечения, программного обеспечения и компонентов микрокода. Поддержка VPD для плат расширения является необязательной в зависимости от производителя. Введение VPD не оказывает влияния на существующие PCI-устройства и требует минимума изменений в устройствах, которые будут содержать VPD. Поддержка VPD для плат расширения является необязательной, но желательной из-за существенных выгод для платы расширения, производителей системных плат и для механизма Plug & Play. Механизм для доступа к VPD и описание соответствующих структур данных приведены в *приложении 3*.

Драйверы устройств

Есть две характеристики PCI-устройств, из-за которых драйверы устройств PCI могут отличаться от "стандарта" или существующих драйверов устройств. Первая характеристика — это способность PCI-устройств перемещаться в адресных пространствах. Драйверы устройств PCI (и другое конфигурационное программное обеспечение) должны использовать информацию отображения, хранимую в регистрах пространства конфигурации устройства, чтобы определить, куда было отображено устройство. То же относится и к определению использования линии прерывания. Второй характеристикой являются общие прерывания PCI. Драйверы устройств PCI обязаны поддерживать разделяемые (общие) прерывания, т. к. вероятно, что в системе будет соединено больше, чем одно устройство с одной линией прерывания. Точный

[7] Разделяется именно дешифратор, а не сами регистры. Регистр базового адреса дополнительного ПЗУ и другие регистры базовых адресов должны содержать различные значения в одно и то же время.

метод для разделения прерываний является спецификой операционной системы. Некоторые системы не могут гарантировать, что данные доставлены в память прежде, чем прерывания доставлены центральному процессору. Такие события могут привести к проблемам последовательности данных (потери данных). Эта ситуация наиболее часто ассоциируется с реализацией буферов отложенной записи в мостах между шиной PCI и другими шинами. Есть три пути (три метода), чтобы гарантировать последовательность данных и прерываний:

1. Аппаратные средства системы могут гарантировать, что буферы отложенной записи очищаются (сбрасываются на диск) прежде, чем прерывания доставляются процессору.
2. Устройство, генерирующее прерывание, может выполнить чтение только что записанных данных перед генерированием прерывания. Это приводит к очищению буферов.
3. Драйвер устройства может выполнить чтение из любого регистра в устройстве перед доступом к данным, записанным устройством. Это чтение приводит к очищению буферов.

Драйверы устройства, в конечном счете, отвечают за гарантию последовательности прерываний и данных в предположении, что, по крайней мере, один из трех методов, описанных выше, реализован в системе. Это означает, что драйвер устройства должен выполнить метод 3, если метод 2 устройства не выполнен или драйвер располагает информацией, что метод 1 выполнен аппаратными средствами системы.

Сброс системы

После сброса системы процессор(ы) должен быть способен получить доступ к загрузочному коду и любым устройствам, необходимым для инициализации и загрузки операционной системы. В зависимости от архитектуры системы, мостам может потребоваться функциональность передачи доступов на отдаленную шину. Точно так же устройствам на PCI может потребоваться способность распознавать фиксированные адреса, чтобы поддержать последовательность загрузки в архитектуре системы. Такие устройства обязаны поддерживать регистр команд. Они должны также обеспечить механизм (вызываемый через пространство конфигурации), чтобы разрешить распознавание фиксированных адресов.

Список функциональностей

Некоторые возможности поддерживаются путем добавления набора регистров к связанному списку, называемому *списком функциональностей* или,

иначе, "Capabilities List". Реализация данной структуры необязательна. На указатель функциональностей или, иначе, регистр Capabilities Pointer, расположенный по смещению 34h, в свою очередь указывает бит **Capabilities List** (бит 4) в регистре статуса PCI, причем бит должен быть установлен. Регистр Capabilities Pointer указывает на первое вхождение в списке. Каждая функциональность в списке состоит из 8-битного поля **ID**, назначенного комитетом PCI SIG, 8-битного указателя на следующую функциональность, и некоторого числа дополнительных регистров сразу же после указателя для ее реализации. Каждая функциональность должна быть выровнена на границу двойного слова. Младшие два бита всех указателей (включая начальный указатель по смещению 34h) зарезервированы и должны иметь значение "00b", хотя программное обеспечение обязано маскировать их, чтобы учесть будущие использования этих битов. Значение указателя "00h" используется, чтобы указать заключительную функциональность в списке. Пример организации списка показан на рис. 11.8.

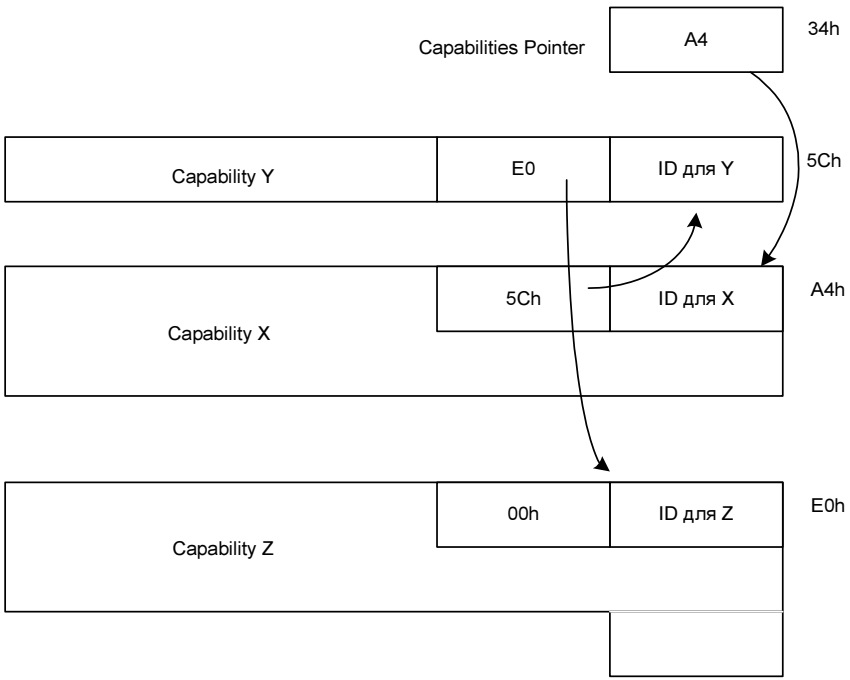


Рис. 11.8. Пример списка функциональностей

Каждая функциональность должна иметь ID-код, назначенный комитетом SIG. Эти коды назначаются и обрабатываются так же, как и коды класса. Для каждой функциональности обязательно определяется детальная карта регист-

ров. Эти регистры должны следовать немедленно за указателем на следующую функциональность. Список определенных в настоящее время функциональностей приведен в *приложении 1*.

Механизм запроса прерывания

Message Signaled Interrupts (MSI) — механизм запроса прерывания через сообщение. Если устройству требуется вызвать прерывание, то оно производит запись определенного формата по определенному адресу (PCI-транзакция записи двойного слова в память). Адрес транзакции определяет место назначения сообщения, и данные транзакции определяют само сообщение. Системное программное обеспечение инициализирует место назначения сообщения и само сообщение во время конфигурирования устройства, выделяя один или более неразделяемых сообщений для каждой MSI-совместимой функции. Поскольку цель транзакции не может отличить транзакцию записи MSI от любой другой транзакции записи, то выполняются все условия завершения транзакций. Поэтому транзакция записи MSI может быть завершена с типом "Retry", "Master-Abort", "Target-Abort" или нормальным завершением. При разработке устройств рекомендуется реализовывать выводы прерываний в целях обеспечения совместимости в системах, не поддерживающих MSI. Предполагается, что через некоторое время потребность в выводах прерываний снизится. Устройства, которые не поддерживают выводы прерываний из-за ограничений количества выводов (и полагаются на опрос для обслуживания устройства), могут реализовать сообщения для увеличения производительности без добавления дополнительных выводов. Поэтому системное конфигурационное ПО при обслуживании устройства, поддерживающее сообщения, должно считать вывод прерывания отсутствующим. Задержка прерывания (время от передачи сигналов прерывания до обслуживания прерывания) зависит от системы. Совместимый с обычными механизмами прерываний механизм сообщения о прерывании не предоставляет гарантий времени задержки прерывания [9].

Структура функциональности MSI

Идентификация и конфигурирование MSI-совместимого устройства осуществляется через механизм функциональностей. Структура функциональности сообщения приведена на рис. 11.9. Каждая функция устройства, которая поддерживает MSI (в многофункциональном устройстве), должна реализовать собственную структуру функциональности MSI. Для каждой функции может быть реализована только одна структура MSI.

Для запроса на обслуживание функция MSI записывает содержимое регистра Message Data по адресу, указанному в регистре Message Address (и дополни-

тельно в регистре Message Upper Address для 64-битного адреса сообщения). Чтение по адресу, указанному в регистре Message Address, выдает неопределенные результаты. Для реализации структуры функциональности для 32-битного адреса сообщения (показанной на рис. 11.9) функция должна поддерживать адрес сообщения такой разрядности. Аналогично для реализации 64-битного адреса. Если устройство поддерживает MSI и 64-битную адресацию (DAC) при действии его в качестве мастера, то оно должно реализовать структуру 64-битного адреса сообщения.

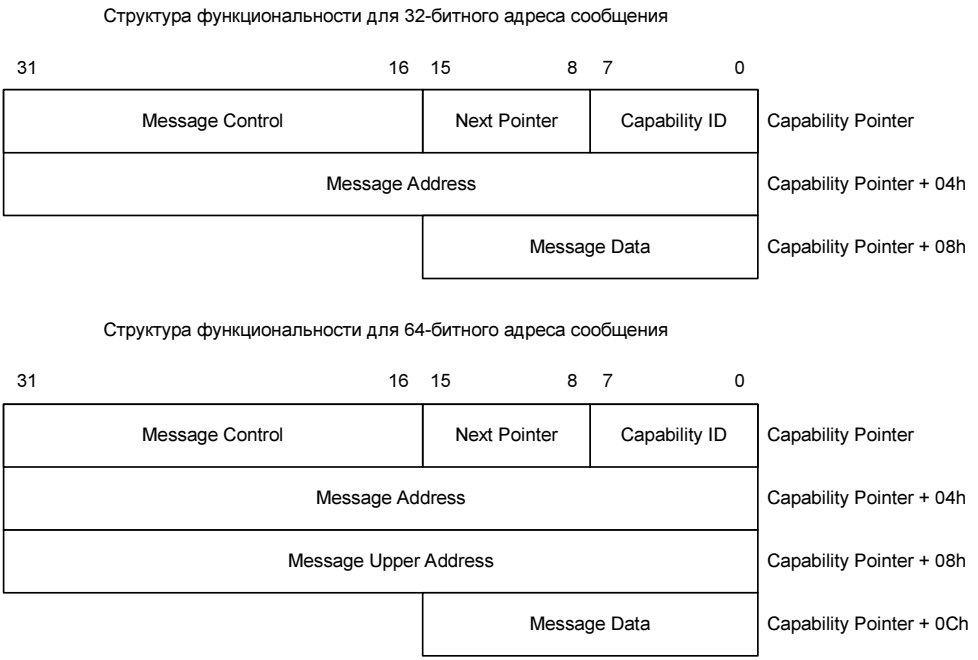


Рис. 11.9. Структура функциональности MSI

Регистр управления сообщением указывает наличие функциональности и обеспечивает системное программное обеспечение контролем над MSI. Каждое поле структуры описано в следующих разделах, номера битов указаны относительно начала регистра. Зарезервированные регистры и биты всегда возвращают "0" при чтении, запись в них не имеет никакого эффекта. Регистры, доступные только для чтения, возвращают действительные данные, если операции чтения и записи не имеют никакого эффекта.

- **Capability ID** (биты 7—0). Сокращенное обозначение "CAP_ID". Значение "05h" данного поля структуры идентифицирует MSI-совместимую функцию. Поле доступно только для чтения.

- **Next Pointer** (биты 15—8). Сокращенное обозначение "NXT_PTR". Указатель на следующий пункт в списке функциональностей. Должен быть нулем для последнего пункта в списке. Поле доступно только для чтения.
- **Message Control**. Этот регистр позволяет системному программному обеспечению управлять MSI. После сброса механизм MSI отключен (бит 0 очищен) и запросы функции обслуживаются через вывод INTx# (если он реализован). Системное ПО может включить MSI, установив бит 0 в этом регистре. Системное ПО также может изменять доступные для чтения/записи биты и поля данного регистра. Драйверу устройства запрещено изменять доступные для чтения/записи биты и поля регистра. Далее описано назначение битов регистра Message Control.
  - *Биты 15—8*. Зарезервированы. Всегда возвращают "0" при чтении, запись не производит никакого эффекта.
  - *Бит 7*. **64-bit Address Capable**. Поддержка 64-битной адресации. Значение "1" означает, что функция поддерживает генерацию 64-битного адреса сообщений, значение "0" — что не поддерживает. Бит доступен только для чтения.
  - *Биты 6—4*. **Multiple Message Capable**. Разрешение нескольких сообщений. Системное ПО производит запись в эти биты, тем самым указывая количество выделенных сообщений (равное или меньшее количеству запрошенных). Количество выделенных сообщений выровнено на степень двойки. Если функция запросила четыре сообщения (указав в битах **Multiple Message Capable** значение "010"), системное ПО может выделить 4, 2 или 1 сообщение, записав в них "010", "001 или "000" соответственно. Если механизм MSI включен, то устройству будет выделено как минимум одно сообщение. Соответствие значения этих битов количеству выделенных сообщений приведено в табл. 11.2.

Таблица 11.2. *Количество выделенных сообщений*

Значение	Количество выделенных сообщений
000	1
001	2
010	4
011	8
100	16
101	32
110	Зарезервировано
111	Зарезервировано

Значение этих битов после сброса будет "000", они доступны для чтения/записи.

- *Биты 3—1*. **Multiple Message Capable**. Системное ПО определяет количество запрошенных сообщений по значению этих битов. Величина должна быть выровнена на степень двойки (если функции требуется три сообщения, она запрашивает четыре, инициализируя поле этих битов значением "010"). Значения поля идентичны приведенным в табл. 11.2 за исключением того, что поле доступно только для чтения.
- *Бит 0.* **MSI Enable**. Значение "1" означает, что функции разрешено использовать MSI для запроса обслуживания и запрещено использовать выводы INTx# (если они реализованы). Системное конфигурационное ПО устанавливает этот бит, чтобы включить MSI. Драйверу устройства запрещено записывать этот бит, чтобы маскировать запрос обслуживания функции. Значение "0" означает, что функции запрещено использовать MSI для запроса на обслуживание. После сброса этот бит установлен в "0" (MSI выключен). Бит доступен для чтения/записи.

- **Message Address** (биты 31—02). Специфический системный адрес сообщения. Если бит **Message Enable** (бит 0 регистра Message Control) установлен, то содержимое этого регистра определяется выровненным на двойное слово адресом (AD[31::02]) для транзакций MSI записи в память. На линиях AD[1::0] должно быть нулевое значение во время адресной фазы. Поле доступно для чтения/записи.
- **Message Address** (биты 1—0). Зарезервированы, всегда возвращают значение "0" при чтении; запись не имеет эффекта.
- **Message Upper Address** (биты 31—00). Специфический системный адрес сообщения, старшая часть. Реализация данного регистра необязательна и требуется, только если устройство поддерживает 64-битный адрес сообщения (установлен бит 7 в регистре Message Control)[8]. Если установлен бит **Message Enable** (бит 0 регистра Message Control), содержимое этого регистра (если оно не нулевое) определяется старшими 32- или 64-битным адресом сообщения (AD[63::32]). Если содержимое этого регистра нуль, то устройство использует 32-битный адрес, определяемый регистром Message Address. Биты доступны для чтения/записи.
- **Message Data** (биты 15—00). Каждой MSI-функции может быть выделено до 32-х уникальных сообщений. Количество поддерживаемых уникальных сообщений определяется системной архитектурой. Если установлен бит

[8] Этот регистр требуется, когда устройство поддерживает 64-битную адресацию при функционировании как мастер.

Message Enable (бит 0 регистра Message Control), то данные для сообщения передаются через младшее слово (AD[15::00]) фазы данных транзакции записи в память. На линиях AD[31::16] должно быть нулевое значение во время фазы данных транзакции записи. Сигналы C/BE[3::0]# выставлены во время фазы данных транзакции записи в память. Биты **Multiple Message Enable** (биты 6—4 регистра Message Control) определяют количество бит данных сообщения, которые функция может модифицировать. Изменяя значения битов, функция устанавливает номер сообщения. Например, значение "010" битов **Multiple Message Enable** указывает функции, что было выделено четыре сообщения и разрешено модифицировать биты данных сообщения 1 и 0 ($4_{10} = 100_2$). Если значение этих битов равно "000", то функции не разрешено изменять данные сообщения. Биты доступны для чтения/записи.

Алгоритм функционирования MSI

Во время конфигурирования системное программное обеспечение проверяет список функциональностей функции. Если по смещению 05h обнаружено поле **Capability ID**, то в функции реализовано MSI. Системное программное обеспечение читает регистр Message Control структуры MSI, чтобы определить возможности функции.

Системное ПО читает поле **Multiple Message Capable** регистра Message Control, чтобы определить число запрошенных сообщений. Далее системное ПО производит запись в поле **Multiple Message Enable** этого же регистра для выделения сообщений. Например, функция может запросить четыре сообщения и ей будет выделено четыре, два или одно сообщение. Число сообщений, запрошенных и выделенных, выравниваются по степени двойки (функция, которой требуется три сообщения, должна запросить четыре). Если бит **64-bit Address Capable** установлен, то программное обеспечение системы инициализирует регистр Message Address и регистр Message Upper Address определяемым системой значением адреса сообщения. Системное программное обеспечение может записать в регистр Message Upper Address ноль и функция сгенерирует 32-битный адрес для транзакций записи MSI. Если же бит **64-bit Address Capable** сброшен, то системное программное обеспечение инициализирует регистр Message Address определяемым системой адресом назначения сообщения. Системное программное обеспечение инициализирует регистр Message Data определяемым системой сообщением. Для поддержки обратной совместимости бит **MSI Enable** регистра Message Control сбрасывается после повторной установки (MSI — выключен). Конфигурационное ПО устанавливает этот бит для включения механизма MSI. Драйверу устройства

запрещено переписывать этот бит для маскирования запроса обслуживания функции. После включения механизма MSI функции запрещено использовать вывод INTx# (если он реализован) для запроса на обслуживание. Механизм MSI и вывод INTx# являются взаимно исключающими. После включения MSI функция может посылать сообщения. Для этого функция выполняет запись в память размером в двойное слово по адресу, определенному содержимым регистра Message Address (для 64-битного адреса дополнительно содержимым регистра Message Upper Address). Записанное двойное слово состоит из значения регистра Message Data в двух младших байтах и нулей в двух старших байтах. Если поле **Multiple Message Enable** регистра Message Control не равно нулю, то устройству разрешено изменить порядок младших битов данных сообщения для генерирования нескольких сообщений. Например, значение поля **Multiple Message Enable** равное "010" указывает, что функции разрешено изменить биты 1 и 0, чтобы сгенерировать до четырех уникальных сообщений. Если же значение поля **Multiple Message Enable** — "000", то функции не разрешают изменить данные сообщения. Как функция использует несколько сообщений — зависит от устройства. Если устройство передает одно и то же сообщение много раз, то будет гарантированно обслужено только одно сообщение. Если должны быть обслужены все сообщения, то требуется связь "рукопожатием" с драйвером устройства. Это означает, что как только функция передает сообщение A, то она не может снова передать сообщение A, пока ей явно не разрешит ее драйвер устройства. Если некоторые сообщения могут быть потеряны, то "рукопожатие" с драйвером устройства не требуется. Обслуживание каждого сообщения гарантируется для устройств, поддерживающих несколько различных сообщений. Например, устройство может передавать сообщение A после сообщения B без вмешательства драйвера устройства. Сообщение A и сообщение B будут обслужены. Механизм MSI по определению является механизмом с не разделяемыми прерываниями и предписывает последовательную передачу данных. Архитектура системы гарантирует, что запись данных до посылки MSI достигает заключительного предназначения прежде, чем процедура обработки прерывания обратится к этим данным. Поэтому драйвер устройства не должен выполнять чтение из устройства перед обслуживанием его MSI.

Завершение транзакции MSI

Цель MSI-транзакции не в состоянии отличить транзакцию MSI от любой другой транзакции записи в память. Требования завершения MSI-транзакции практически не отличаются от требований завершения обычных транзакций записи в память, кроме незначительных дополнений. Если транзакция записи MSI завершена с типом "Master-Abort" или "Target-Abort", мастер транзакции

записи в память MSI обязан сообщить об ошибке установкой сигнала SERR# (если установлен бит 8 в регистре Command) и установить соответствующие биты в регистре Status. Транзакция записи в память MSI игнорируется целью, если она завершена с типом "Master-Abort" или "Target-Abort". Генерирование сигнала SERR# в MSI-совместимом окружении, содержащем мосты PCI-to-PCI, требует, чтобы биты разрешения сообщения об ошибках по линии SERR# во всех устройствах на маршруте MSI-сообщения были установлены. Подробный протокол сообщения по линии SERR# мостов PCI-to-PCI изложен в спецификации "PCI-to-PCI Bridge Architecture Specification". Если транзакции записи MSI приводят к ошибке четности данных, то мастер транзакции записи MSI должен выставить сигнал SERR# (если установлен бит 8 в регистре Command) и установить соответствующие биты в регистре Status.

Требования приема и упорядочивания MSI-транзакций

Транзакция MSI является транзакцией записи в память. Как и для обычных транзакций записи в память, устройство, содержащее цель сообщения прерываний (приемник прерывания), обязана выполнять все транзакции MSI как цель, не запрашивая других транзакций для выполнения их в качестве мастера. Это означает, что приемник сообщения должен выполнить транзакцию MSI, независимо от того, когда центральный процессор обслужит прерывание. Например, каждый раз при получении приемником прерывания сообщения он может устанавливать бит во внутреннем регистре. Таким способом приемник указывает, что сообщение было получено и затем выполняет транзакцию на шине. Соответствующая процедура обслуживания прерывания была бы вызвана позже, потому был установлен этот бит. Приемнику сообщения не разрешено задерживать выполнение сообщения о прерывании на шине, ожидая подтверждения от процессора, что прерывание было обслужено. Это объясняется тем, что подобная зависимость может привести к "зависанию" шины, если несколько устройств выполняют сообщения о прерывании одновременно. Хотя сообщения остаются строго упорядоченными везде в иерархии шины PCI, порядок получения сообщений о прерывании не гарантирует точно такого же порядка обслуживания прерываний. Поскольку приемник сообщения должен выполнить все транзакции MSI независимо от фактического обслуживания прерывания, то приемник не будет сохранять порядок, в котором были получены прерывания. Данное правило выполняется для сообщений о прерывании, полученных от различных устройств и для сообщений, полученных от одних и тех же устройств. Если устройству требуется обслуживание одного прерывания раньше другого, то устройство не должно посылать второе сообщение о прерывании, пока не будет обслужено первое.

ПЗУ расширения

Устройствам может требоваться дополнительное ПЗУ для хранения и выполнения специфических функций инициализации или загрузки системы. Спецификация PCI поддерживает *дополнительные ПЗУ*, называемые иначе как *ПЗУ расширения* или Expansion ROM. Правила организации позволяют ПЗУ содержать несколько различных образов, совместимых с различными машинами и архитектурами процессоров. В данном разделе описано расположение кода в ПЗУ расширения и назначение полей заголовков. PCI-устройства, реализующие эту возможность, должны обеспечить обращение к ПЗУ расширения с любой комбинацией байта разрешения. Это означает, что должны поддерживаться доступы к ПЗУ расширения размером в двойное слово. Информация, содержащаяся в ПЗУ, совместима с существующим заголовком ПЗУ расширения ISA для архитектуры Intel *x*86, но также поддерживаются другие архитектуры. Структура заголовка была определена таким образом, чтобы во время выполнения кода использовалось минимальное количество памяти и максимально эффективно выполнялась функция платы расширения [9].

Информация заголовка строится в соответствии со следующими правилами:

- размер кода указывает полное смежное адресное пространство, необходимое образу PCI ПЗУ во время инициализации;
- тип выполнимого или интерпретируемого кода определяется специальным идентификатором;
- предоставляется информация о версии кода и данных ПЗУ;
- значения "Vendor ID" и "Device ID", назначенные PCI SIG данному устройству, содержатся в ПЗУ.

Код ПЗУ (ROM) никогда не выполняется внутри самой микросхемы. Он всегда копируется из микросхемы в память и выполняется непосредственно в памяти. За счет этого обеспечивается динамическое изменение размера кода и повышение скорости выполнения самого кода.

Состав ПЗУ расширения

ПЗУ расширения могут содержать выполнимый или интерпретируемый код для нескольких процессорных архитектур. Физически код размещается в одном устройстве ПЗУ или ППЗУ, при этом количество образов для различных систем и архитектур не ограничено. Структура ПЗУ расширения представлена на рис. 11.10. Каждый образ должен начинаться на 512-байтной границе и содержать заголовок определенного формата. Стартовая точка каждого образа зависит от размера предыдущих образов. Последний образ в ROM содержит в заголовке специальное значение — признак, чтобы идентифицировать его как последний образ [9].

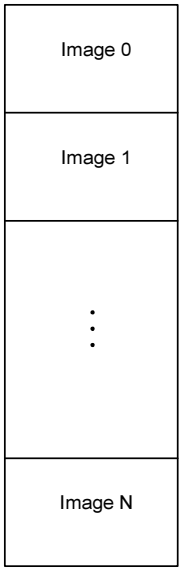


Рис. 11.10. Структура ПЗУ расширения PCI-спецификации

Формат заголовка

Информация в каждом образе разделена на две различные области. Первая область, заголовок ПЗУ (ROM header), должна быть расположена в начале образа, вторая область, структура данных PCI (PCI Data Structure), должна располагаться после заголовка в первых 64 Кбайтах образа. Формат заголовка приведен в табл. 11.3, назначение полей описано далее. Смещение отсчитывается от начала образа, длина каждого поля приведена в байтах. Все значения указаны в шестнадцатеричном виде. В зависимости от специфики архитектуры к заголовку ПЗУ расширения PCI-спецификации и/или PCI-структуре данных могут быть добавлены дополнительные поля.

Таблица 11.3. *Формат заголовка ПЗУ расширения PCI-спецификации*

Смещение	Длина, байт	Значение	Обозначение
0h	1	55h	ROM Signature
1h	1	AAh	
2h—17h	16	XX	Reserved (Зарезервировано) (данные, уникальные для процессорной архитектуры)
18h—19h	2	XX	Pointer to PCI Data Structure

- **ROM Signature**. Двухбайтное поле, содержащее значение 55h в первом байте и AAh во втором байте. Эта сигнатура должна присутствовать в адресном пространстве ПЗУ для каждого образа.
- **Pointer to PCI Data Structure**. Указатель на структуру данных PCI, а именно двухбайтное значение в формате "little-endian", которое указывает на структуру "PCI Data Structure". Адрес, хранимый в этом указателе, является началом образа ПЗУ.

Формат структуры данных PCI

Структура данных PCI (PCI Data Structure) располагается в пределах первых 64 Кбайтах образа ПЗУ и должна быть выровнена на границу двойного слова. Структура содержит информацию, приведенную в табл. 11.4.

Таблица 11.4. *Формат структуры данных PCI*

Смещение	Длина, байт	Описание
0	4	Signature = "PCIR"
4	2	Vendor Identification
6	2	Device Identification
8	2	Reserved
A	2	PCI Data Structure Length
C	1	PCI Data Structure Revision
D	3	Class Code
10	2	Image Length
12	2	Revision Level of Code/Data
14	1	Code Type
15	1	Indicator
16	2	Reserved

- **Signature**. Поле размером четыре байта представляет уникальную сигнатуру для структуры данных PCI. Строка "PCIR" обозначает сигнатуру, при этом "P" начинается со смещения 0, "C" начинается по смещению 1 и т. д.
- **Vendor Identification**. Поле размером 16 бит, содержит значение, идентичное значению, содержащемуся в поле **Vendor ID** пространства конфигурации для этого устройства.

- **Device Identification**. Поле размером 16 бит, содержит значение, идентичное значению, содержащемуся в поле **Device ID** пространства конфигурации для этого устройства.
- **Reserved**. Зарезервированные 16-битные поля. В устаревших версиях спецификации PCI это поле указывало на размещенные в ПЗУ (ROM) структуры данных "Vital Product Data".
- **PCI Data Structure Length**. 16-битное поле, которое определяет длину структуры данных. Длина измеряется от начала структуры данных (первый байт поля **Signature**). Значение этого поля хранится в формате "little-endian" и указано в байтах.
- **PCI Data Structure Revision**. Поле размером 8 бит, идентифицирует версию структуры данных. Для спецификации 2.3 номер версии равен 0.
- **Class Code**. Поле размером 24 бита, содержит значение, идентичное значению, содержащемуся в поле **Class Code** пространства конфигурации для этого устройства.
- **Image Length**. Поле указывает длину образа. Значение данного поля хранится в формате "little-endian" и измеряется в единицах по 512 байт. Размер поля составляет 2 байта.
- **Revision Level**. Поле содержит версию кода в образе ПЗУ (ROM). Размер поля 2 байта.
- **Code Type**. Однобайтное поле, которое идентифицирует тип кода, содержащийся в этой секции ПЗУ (ROM). Код может быть исполнимым двоичным кодом для определенного процессора и системной архитектуры или же интерпретируемым кодом. Типы кодов определены в табл. 11.5.

Таблица 11.5. *Типы поддерживаемых кодов*

Тип	Описание
0	Intel *x*86, PC/AT
1	Open Firmware PCI[9]

[9] Open Firmware является независимым стандартом процессорной и системной архитектуры для обработки специфического кода дополнительного ПЗУ какого-либо устройства. Документация на данный стандарт доступна в *IEEE 1275-1994 Standard for Boot (Initialization, Configuration) Firmware Core Requirements and Practices.* Связанный с ним документ, *PCI Bus Binding to IEEE 1275-1994*, определяет применение Open Firmware для шины PCI, включая специфические требования и правила техники эксплуатации для PCI. Данный документ может быть получен, используя анонимное соединение по FTP, по адресу **playground.sun.com**, файл **/pub/p1275/bindings/postscript/PCI.ps**.

Таблица 11.5 *(окончание)*

Тип	Описание
2	Hewlett-Packard PA RISC
3	Extensible Firmware Interface (EFI)
4—FF	Зарезервированные

- **Indicator**. Значимым битом в данном поле является бит 7, это признак заключительного образа в ПЗУ. Значение "1" обозначает "последний образ", а значение "0" указывает, что существует следующий образ. Биты 0—6 зарезервированы.

Код процедуры POST

В большинстве случаев системный код процедуры POST (Power-on Self Test) обрабатывает PCI-устройства платы расширения так же, как и устройства, находящиеся на системной плате. Исключение составляет обработка устройств ПЗУ расширения. Код процедуры POST обнаруживает присутствие ПЗУ расширения (Expansion ROM) в два этапа. Сначала код определяет, реализован ли устройством регистр Expansion ROM Base Address в пространстве конфигурации. Если регистр реализован, то процедура POST должна отобразить и включить ПЗУ в неиспользованную часть адресного пространства и проверить первые два байта на наличие сигнатуры 55AAh. Если сигнатура обнаружена, то ПЗУ присутствует; в противном случае устройство не содержит образа ПЗУ (ROM).

Если ПЗУ присутствует, то процедура POST должна искать образ, содержащий надлежащий тип кода для данного процессора, чьи поля **Vendor ID** и **Device ID** совпадают с соответствующими полями устройства.

После обнаружения надлежащего образа процедура POST копирует требуемые данные в память. После этого выполняется код инициализации устройства. Количество копируемых данных и выполнение кода инициализации устройства зависит от значения в поле **Code Type** структуры данных PCI [9].

Совместимость со стандартом PC

В данном разделе описаны подробные требования и порядок обработки образов ПЗУ (ROM), использующихся в PC-совместимых системах. Данные требования применимы к любому образу, который определен как Intel *x*86, PC/AT-совместимый в поле **Code Type** структуры данных PCI, и к любой PC-совместимой платформе.

Дополнительные поля заголовка

Стандартный заголовок для PCI-образов (табл. 11.6) имеет два дополнительных поля для PC-совместимости. Первое дополнительное поле, по смещению 02h, содержит значение размера кода инициализации для образа. Второе значение, находящееся по смещению 03h, представляет собой точку входа для функции `INIT`.

Таблица 11.6. *Формат PC-совместимого заголовка*

Смещение	Длина, байт	Значение	Описание
0h	1	55h	ROM Signature
01h	1	AAh	
02h	1	XX	Initialization Size — размер кода в единицах по 512 байт
03h	3	XX	Точка входа для функции `INIT`. Процедура POST выполняет дальний вызов `FAR CALL` по указанному адресу
06h—17h	12	XX	Зарезервировано
18h-19h	2	XX	Указатель на структуру "PCI Data Structure"

Алгоритм функционирования POST при обработке образа

Код POST в PC-совместимых системах копирует в оперативную память (RAM) количество байтов, указанных в поле **Initialization Size**, и затем вызывает функцию `INIT`, точка входа которой указана значением по смещению 03h. Процедура POST должна копировать код в перезаписываемую область RAM, или пометить ее как перезаписываемую на время выполнения кода расширения. Это позволяет функции `INIT` хранить некоторые статические данные в области оперативной памяти (RAM) и регулировать размер во время выполнения кода таким образом, чтобы он потреблял меньше места во время функционирования системы. Алгоритм функционирования процедуры POST для PC-совместимых систем для обработки каждого расширения ПЗУ состоит из следующих этапов:

1. Отображение и включение ПЗУ расширения в незанятую область адресного пространства памяти.
2. Поиск надлежащего образа в ПЗУ и копирование его в определенную стандартом "PC Specification" область памяти (обычно диапазон физиче-

ских адресов от 0C0000h до 0DFFFFh), используя значение поля **Initialization Size**.

3. Отключение регистра Expansion ROM Base Address.
4. Разрешение перезаписи для области памяти и вызов функции INIT.
5. Определение необходимого количества памяти для выполнения кода. Для этого используется значение байта по смещению 02h (которое, возможно, было изменено функцией INIT).

Перед системной загрузкой код процедуры POST должен запретить запись в область RAM, содержащую код ПЗУ расширения. Обработка процедурой POST устройств VGA несколько отличается. Код BIOS VGA-устройства должен быть скопирован по адресу 0C0000h. Идентификация VGA-устройств осуществляется через поле **Class Code** в пространстве конфигурации устройства.

Расширения функции *INIT*

На функцию INIT, содержащуюся в ПЗУ расширения, возлагается ответственность за инициализацию устройства ввода-вывода и подготовку к переводу его в рабочее состояние. Поскольку в область памяти, где расположен код, разрешена запись, то функция INIT может выполнять более широкий класс операций.

Одной из таких возможностей является хранение статических данных в памяти во время выполнения функции INIT. Эти данные впоследствии могут использоваться кодом BIOS или драйверами устройства. После возврата из функции INIT область памяти будет недоступна для записи.

Функция INIT может изменять размер памяти, требуемой во время выполнения, путем модификации байта размера по смещению 02h в образе. Это позволяет не выходить за пределы ограниченного ресурса памяти в области 0C0000h—0DFFFFh. Например, дополнительное ПЗУ устройства может требовать 24 Кбайта для его кода инициализации и исполняемого кода, при этом для исполняемого кода требуется только 8 Кбайт. В образе ПЗУ указано значение 24 Кбайт, чтобы процедура POST копировала все это в память. Функция INIT во время выполнения может изменить байт размера до величины 8 Кбайт. Когда функция INIT возвращает управление, процедура POST обнаруживает, что размер исполняемого кода 8 Кбайт и может копировать следующий образ по оптимальному адресу. Функция INIT должна гарантировать правильность контрольной суммы для всего образа. Если функция INIT изменяет область памяти любым способом, то должна быть рассчитана новая контрольная сумма и сохранена в образе. Функция INIT не должна изменять па-

мять системы (исключая область памяти функции INIT) любым способом, если она не использует соответствующий протокол или сервисы BIOS для выделения памяти. Нередко процедуры POST используют память системы для критических данных или кода, и его разрушение или модификация может препятствовать загрузке системы. Для удаления функции INIT из области памяти ПЗУ в поле **Initialization Size** записывается ноль. В этом случае контрольная сумма не вычисляется (поскольку размер области для контрольной суммы равен нулю). На входе функции INIT передаются три параметра: номер шины, номер устройства и номер функции устройства, реализовавшего ПЗУ расширения. Эти параметры могут использоваться для получения доступа к устройству, которое должно быть инициализировано. Их передают через регистры *x*86, регистр AH содержит номер шины, старшие пять битов регистра AL содержат номер устройства, и младшие три бита регистра AL содержат номер функции. До вызова функции INIT процедура POST должна выделить ресурсы устройству (через регистры Base Address и Interrupt Line) [9].

Структура образа

Структура PC-совместимого образа, представленная на рис. 11.11, определяется тремя величинами:

- размером исполнимого кода;
- размером кода инициализации;
- размером образа.

Размер образа — полная длина образа, и она должна быть больше или равняться размеру кода инициализации.

Размер кода инициализации определяет количество образа, которое содержит код инициализации и исполнимый код. Это то количество данных, которые процедура POST копирует в память перед выполнением процедуры инициализации. Размер кода инициализации должен быть больше или равен размеру исполняемого кода. Данные инициализации, которые скопированы в память, должны в сумме с контрольной суммой давать 0 (использование стандартного алгоритма).

Размер исполняемого кода определяет количество образа, которое содержит исполняемый код. Это то количество данных, которые процедура POST оставит в памяти во время работы системы. Контрольная сумма этого количества образа также должна сводиться к нулю.

Структура данных PCI должна содержаться в пределах исполняемого кода в образе (в любом его месте) или в пределах кода инициализации [9].

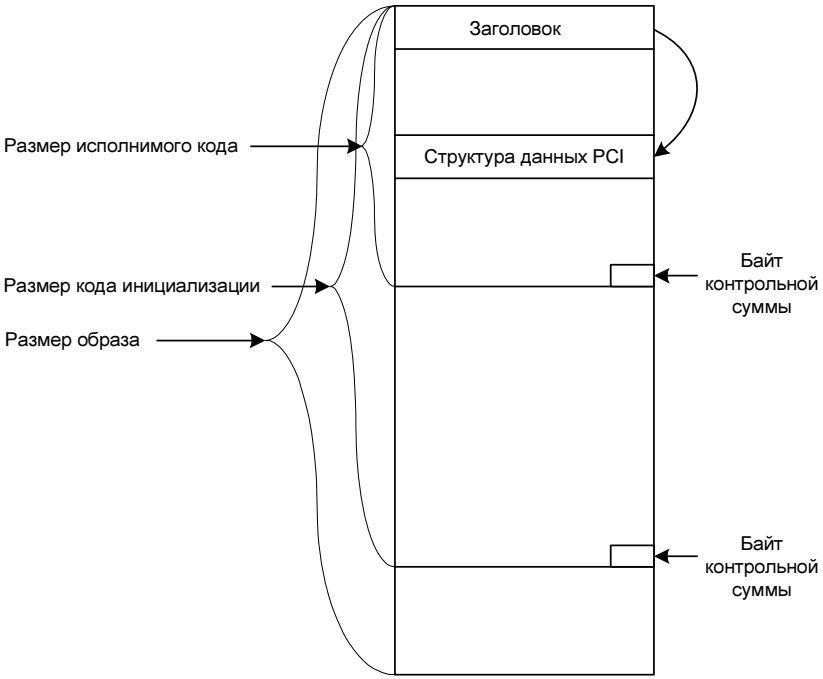
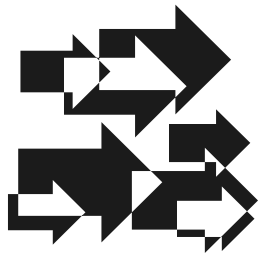


Рис. 11.11. Структура PC-совместимого образа

ГЛАВА 12



Технология PCI Express

Введение

Шина PCI Express, уже реализованная в новых чипсетах Intel, призвана заменить шину PCI и взять на себя задачу по связи компонентов внутри компьютера на ближайшие десять лет. Ранее шина была известна как шина ввода-вывода третьего поколения (3rd Generation I/O, 3GIO). Основным направлением разработки являлось следующее: применение на множестве сегментов рынка, в роли единой универсальной архитектуры ввода-вывода для настольных ПК, мобильных ПК, серверов, рабочих станций и встроенных устройств. Стоимость реализации должна быть равной или меньшей стоимости PCI. Последовательная шина требует наличия меньшего количества линий, а следовательно, обеспечивает более эффективное использование печатной платы.

Шина PCI Express разрабатывалась с учетом существующих стандартов и операционных систем и поддерживает совместимость с шиной PCI на программном уровне, а конфигурирование и драйверы устройств совместимы с существующими PCI-реализациями. Для поддержки PCI Express не потребуется модификации или дополнительного обновления операционных систем. Добавлена поддержка одноранговой связи между двумя или большим количеством иерархических структур [7].

Третье поколение соединений ввода-вывода

Постоянно увеличивающаяся скорость передачи данных, развитие новых стандартов и архитектур, а также огромное количество уже разработанных устройств: электронных компонентов, системных плат и плат расширения — все это предъявляет высокие и труднореализуемые требования к соединениям ввода-вывода третьего поколения. Краткий и не полный перечень требований

показывает факторы, которые необходимо учитывать при разработке новых стандартов вообще и соединений ввода-вывода в частности.

- Поддержка всех сегментов рынка и появляющихся приложений. Единая универсальная архитектура ввода-вывода для настольных ПК, мобильных, рабочих станций, серверов, коммуникационных платформ и встроенных устройств.
- Низкая стоимость реализации — стоимость на системном уровне не должна превышать стоимость PCI.
- Поддержка соединений между различными платформами. Должна быть обеспечена совместимость межкомпонентных и межплатных соединений через разъем или кабель.
- Новые механические форм-факторы. Мобильный, PCI-совместимый и модульный (катриджный) форм-фактор.
- Совместимая с PCI программная модель. Возможность нумерации и конфигурирования аппаратных средств PCI Express за счет использования реализаций системного программного обеспечения PCI без модификаций. Загрузка существующих операционных систем без модификаций. Поддержка существующих драйверов ввода-вывода без их изменений. Возможность конфигурирования/включения новых функциональных возможностей PCI Express путем адаптации конфигурационного механизма PCI.
- Производительность. Соединения с малым временем задержки для максимизации полосы пропускания полезной нагрузки приложения и эффективности канала. Минимизация количества выводов за счет высокой пропускной способности каждого вывода.
- Дополнительные возможности. Поддержка различных типов данных и правил упорядочивания. Управление и распределение питанием. Идентификация возможностей управления питанием данной функции. Возможность перехода функции в определенное состояние потребления питания. Передача сообщения о текущем состоянии питания. Возможность распространения пробуждающих событий. Поддержка качества обслуживания (QoS).
- PCI-совместимая обработка ошибок.

Разработчиками PCI Express введено новое понятие "канала". Под *каналом* понимается двойной однонаправленный канал связи между двумя компонентами. Канал PCI Express состоит из двух отдельно управляемых сигнальных пар с низким напряжением: передающей и приемной пары [7].

Перечислим основные характеристики и свойства канала.

- Основной канал связи — PCI Express Link — состоит из двух однонаправленных каналов, реализованных как передающая пара и принимающая па-

ра. При этом используется код 8b/10b, как обеспечивающий наибольшую скорость передачи данных.

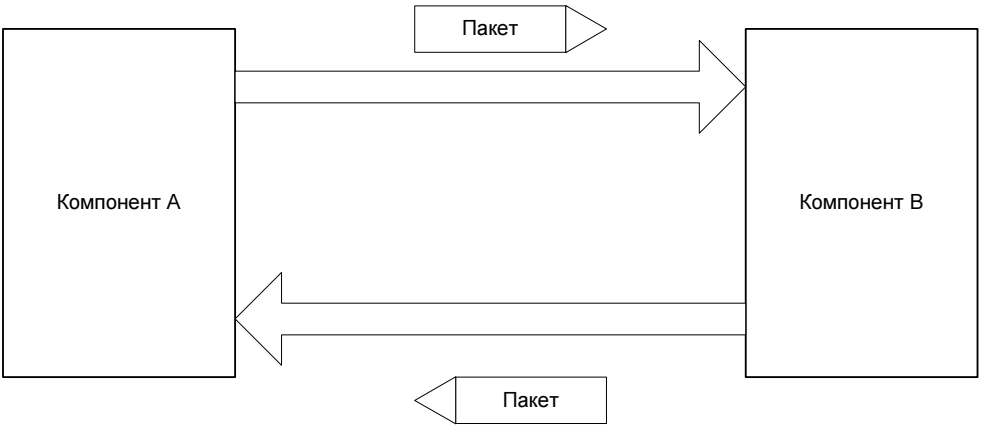


Рис. 12.1. Канал PCI Express

- Сигнальная скорость. После инициализации каждый канал должен функционировать только на одном из поддерживаемых сигнальных уровней. Для версии спецификации 1.0 определена только одна сигнальная скорость, которая обеспечивает максимальную полосу пропускания 2,5 Гбит/с. В дальнейшем предполагается увеличение скорости передачи данных за счет развития технологии.
- Линия. Совокупность различных сигнальных пар объединяется в логическое понятие *линии*. Канал должен поддерживать как минимум одну линию. Для масштабирования полосы пропускания канал связи может группировать несколько линий, обозначаемых как xN, где N является одной из поддерживаемых скоростей канала. Например, канал x8 позволяет сгруппировать полосу 20 Гбит/с в каждом направлении.
- В версии спецификации 1.0 физический уровень поддерживает размерность линий x1, x2, x4, x8, x12, x16, и x32.
- Инициализация. Во время аппаратной инициализации каждый канал PCI Express Link устанавливается в соответствии с размерами линий и частотой функционирования двумя агентами на каждом конце канала. Для этого процесса не требуется программное обеспечение или операционная система.
- Симметрия. Каждый канал должен поддерживать симметричное количество линий в каждом направлении, т. е. канал x16 указывает, что в каждом направлении используются 16 разных (приемных и передающих) сигнальных пар.

Топология PCI Express

Система, построенная на PCI Express, основана на двухточечных каналах, которые соединяют массив компонентов — пример такой системы приведен на рис. 12.2. Данный рисунок иллюстрирует так называемую иерархию, состоящую из компонентов:

- Root Complex (RC) — корневого комплекса;
- Endpoint — оконечного устройства;
- Legacy Endpoint — оконечное устройство, конфигурационный заголовок которого может принадлежать только к одному типу — "00". Под такими устройствами понимаются обычные PCI-устройства. Данный тип введен для обратной совместимости со спецификацией PCI;
- Switch — коммутатора;
- Bridge PCI Express-to-PCI — моста между шинами PCI Express и PCI.

Все эти компоненты соединяются посредством каналов PCI Express Link.

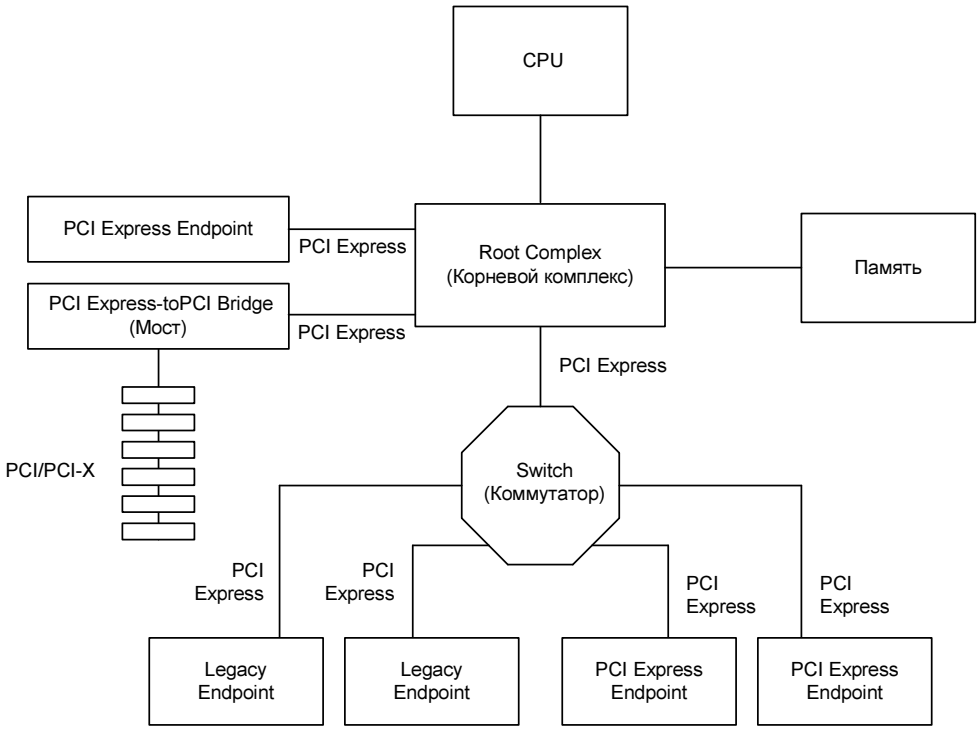


Рис. 12.2. Пример системы на основе PCI Express

Каждый из компонентов отображается в единственное плоское адресное пространство и может быть адресован через PCI-совместимый доступ.

Root Complex

Root Complex (RC) — корневой комплекс — означает главный компонент иерархии ввода-вывода, который соединяет процессор (CPU — central processing unit) и память с устройствами ввода-вывода.

Root Complex может поддерживать один или больше портов PCI Express. Каждый интерфейс определяет отдельную иерархическую группу (домен) ввода-вывода. Каждый иерархический домен может состоять из одного оконечного элемента ввода-вывода — I/O Endpoint — или следующего уровня иерархии, состоящего из одного или более компонентов Switch и I/O Endpoint.

Поддержка передачи одноранговых транзакций между иерархическими доменами через корневой комплекс Root Complex является необязательной и зависит от реализации. Например, возможно объединение реальной или виртуальной коммутации внутренне по отношению к Root Complex для того, чтобы включить полную поддержку одноранговых передач прозрачно для программного обеспечения. Root Complex должен поддерживать генерацию конфигурационных запросов как инициатор запроса, также разрешена генерация запросов ввода-вывода в качестве инициатора запросов. Кроме того, Root Complex не должен поддерживать монопольный доступ как исполнитель, разрешена генерация монопольных доступов в качестве инициатора.

Endpoint

Endpoint — оконечное устройство — означает тип устройства, которое может быть запросчиком или исполнителем транзакций PCI Express как от собственного лица, так и от лица другого, устройства не PCI Express (не имеется в виду устройство PCI или процессор), например, графический контроллер или интерфейс PCI Express-to-USB. Устройства разделяются на два типа: существующие (унаследованные) оконечные устройства и оконечные устройства PCI Express. Для каждого из типов определены отдельные требования.

Правила для существующих оконечных устройств

- Устройство должно иметь заголовок типа "00h" конфигурационного пространства.
- Устройство должно поддерживать конфигурационные запросы как исполнитель.
- Устройство должно поддерживать запросы ввода-вывода как исполнитель.

- Устройство может генерировать запросы ввода-вывода.
- Устройство может поддерживать семантику блокировки памяти как исполнитель, если это необходимо для поддержки программного обеспечения устройства.
- Устройство не должно генерировать блокированные запросы.

Правила для оконечных устройств PCI Express

- Устройство должно иметь заголовок типа "00h" конфигурационного пространства.
- Устройство должно поддерживать конфигурационные запросы как исполнитель.
- Устройство не должно требовать ресурсов ввода-вывода через регистры базовых адресов (BAR — Base Address Registers).
- Устройство не должно генерировать запросов ввода-вывода.
- Устройство не должно поддерживать блокированные запросы как исполнитель или генерировать их как запросчик. Программные драйверы и приложения, совместимые с PCI Express, должны быть разработаны с учетом предотвращения использования семантики блокировки при обращении к оконечному устройству PCI Express.

Switch

Switch — коммутатор определен как логическая совокупность нескольких "виртуальных" мостов PCI-to-PCI, как это показано на рис. 12.3. Все коммутаторы подчиняются следующим основным правилам (расширенные компоненты коммутатора будут поддерживать дополнительные возможности, не описанные здесь).

- Для конфигурационного ПО коммутатор представляет собой один или более логических мостов PCI-to-PCI.
- Коммутатор передает транзакции, используя механизмы PCI-моста, например, адресную маршрутизацию.
- Коммутатор может передавать только двухточечные транзакции между двумя нижерасположенными портами.
- За некоторым исключением коммутатор должен передавать все типы TLP (Transaction Layer Packet) между любыми группами портов.
- Коммутатор должен поддерживать монопольный доступ, поддержка нижележащих портов в качестве инициаторов не требуется.

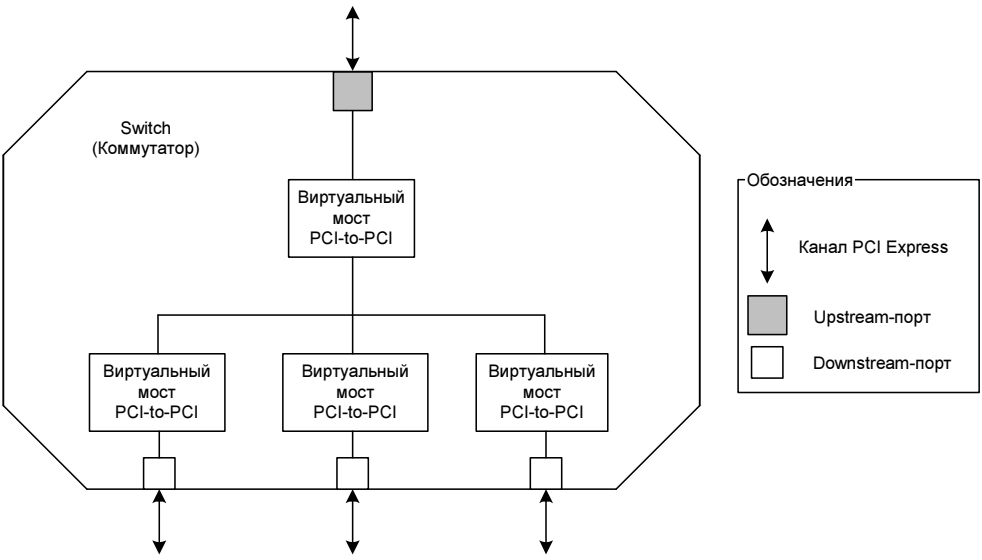


Рис. 12.3. Логическая структура коммутатора

- Коммутатору запрещено разбивать пакеты на пакеты меньшего размера, например, один пакет с полезными данными размером 256 байт не должен быть разделен на два пакета с 128 байтами данных каждый.
- Арбитраж между входными портами коммутатора может быть реализован через круговую систему или взвешенную круговую систему.

Мост PCI Express-to-PCI

Мост между шинами PCI Express и PCI/PCI-X имеет один порт PCI Express и один или много шинных интерфейсов PCI/PCI-X и отвечает следующим требованиям:

- мост PCI Express-to-PCI/PCI-X должен поддерживать все необходимые типы транзакций PCI и/или PCI-X на его интерфейсе PCI;
- мост должен поддерживать монопольный доступ. Мост PCI Express-to-PCI не должен генерировать (или передавать) монопольные доступы от шины PCI к шине PCI Express, но для предотвращения "зависания" должен поддерживать монопольные доступы от шины PCI Express к шине PCI;
- порт PCI Express моста PCI Express-to-PCI должен быть выполнен в соответствии с требованиями поточного управления;
- порт PCI Express моста PCI Express-to-PCI должен быть выполнен в соответствии с требованиями интеграции данных канального уровня.

Обзор уровней PCI Express

Спецификация PCI Express определяет архитектуру в терминах трех раздельных логических уровней:

- ❒ уровня транзакций;
- ❒ канального уровня;
- ❒ физического уровня.

Каждый из уровней разделен на две секции (рис. 12.4), одна из которых функционирует как выходная (передающая информацию), а другая как входная (принимающая информацию). Разделение на уровни чисто логическое и не предполагает детальной реализации в PCI Express [7].

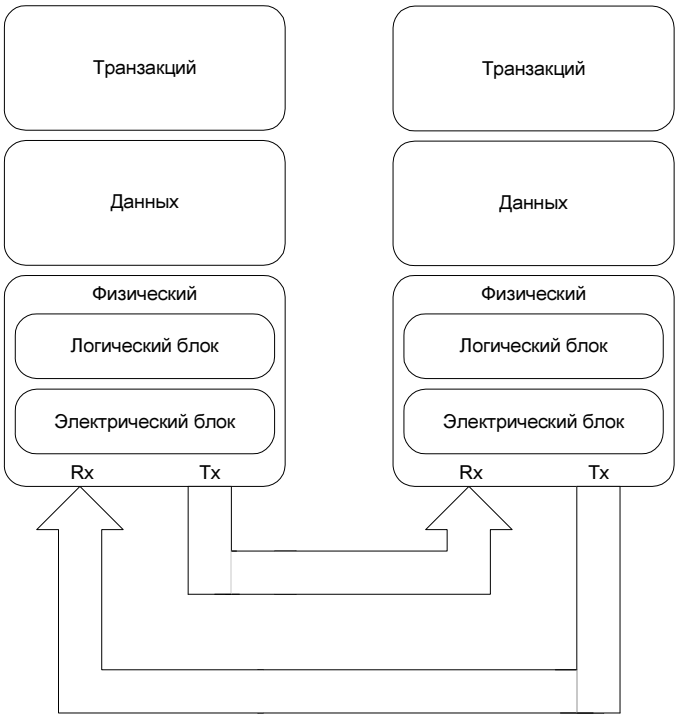


Рис. 12.4. Логические уровни PCI Express

Для обмена информацией между компонентами используются пакеты. Пакеты формируются на уровне транзакций и канальном уровне, чтобы перенести информацию от передающего к принимающему компоненту. Поскольку пакеты передаются через остальные уровни, они дополняются вспомогательной информацией, необходимой для обработки пакета на соответствующем уров-

не. На принимающей стороне происходит обратный процесс, и пакет преобразовывается обратно, начиная с физического уровня, затем на канальном уровне и до уровня транзакций до формата, в котором он может быть обработан принимающим устройством. На рис. 12.5 показан состав пакета для каждого из уровней.

Рис. 12.5. Структура пакета для каждого уровня

Пакетная связь начинается на канальном уровне для реализации функций управления каналом.

Уровень транзакций

Верхним уровнем архитектуры является уровень транзакций. Основная задача данного уровня состоит в сборке/разборке *пакетов уровня транзакций — Transaction Layer Packets (TLP)*. TLP используются для передачи транзакций, таких как чтение и запись, а также как точный тип событий. Уровень транзакций также должен управлять потоком для TLP.

Каждый запрошенный пакет требует ответного пакета, что называется *расщепленные транзакции*. Каждый пакет имеет уникальный идентификатор, который позволяет ответному пакету направляться к правильному инициатору. Формат пакетов поддерживает различные формы адресации (памяти, ввода-вывода, конфигурационные и сообщений) в зависимости от типа транзакции. Пакеты могут также иметь атрибуты, такие как "no-snoop" — не отслеживаемый и "relaxed-ordering" — ослабленное упорядочивание, которые могут быть использованы для оптимизации маршрута этих пакетов при прохождении через систему.

Уровень транзакций поддерживает четыре адресных пространства: три адресных пространства PCI (памяти, ввода-вывода и конфигурационное) и новое пространство сообщений. В спецификации 1.0 концепция MSI (Message Signaled Interrupt) используется как основной механизм прерываний. Пространство сообщений реализовано для поддержки всех предшествующих вспомогательных сигналов, таких как сигналы прерываний, запросы управ-

ления питанием. Транзакции сообщений PCI Express могут быть представлены как "виртуальные провода", так они устраняют широкий ряд вспомогательных сигналов, использующихся в современных реализациях.

Канальный уровень

Средний уровень в стеке — канальный уровень — функционирует как промежуточный этап между физическим уровнем и уровнем транзакций. В функции канального уровня входят управление каналом, обнаружение и исправление ошибок. Принимающая сторона канального уровня принимает пакет TLP (Transaction Layer Packet), собранный на уровне транзакций, вычисляет и вставляет код защиты данных и номер очередности пакета TLP и затем предоставляет пакет на физический уровень для передачи на другую сторону канала. Принимающая сторона канального уровня должна проверить целостность принятого пакета TLP и передать его на уровень транзакций для последующей обработки. При обнаружении ошибки(ок) пакета TLP этот уровень должен запрашивать повторную передачу, пока не будет принята корректная информация, или канал будет определен как сбойный. Канальный уровень также генерирует и принимает пакеты, используемые для функций управления канальным уровнем. Для того чтобы отличить эти пакеты от пакетов, используемых на уровне транзакций (TLP), при генерации и приеме пакетов канального уровня используется термин *Data Link Layer Packet (DLLP)*.

Физический уровень

Физический уровень содержит все необходимые схемы для интерфейсных операций, включая драйвер и входные буферы, параллельное—последовательное и последовательное—параллельное преобразование, схему(ы) ФАПЧ и схему согласования импенданса. Кроме того, он включает также логические функции, связанные с инициализацией и поддержкой интерфейса. Физический уровень обменивается информацией с канальным уровнем в формате, который зависит от реализации. Этот уровень отвечает за преобразование информации, принятой от канального уровня в соответствующий последовательный формат и передачу его через канал PCI Express на частоте и полосе, совместимой с удаленным устройством. Архитектура PCI Express предусматривает поддержку будущего прироста производительности за счет усовершенствования скорости и расширенной техники дешифрации. Будущие скорости и технология дешифрации могут влиять только на физический уровень.

Программная инициализация и конфигурирование

Усовершенствованный механизм обеспечивает увеличение размера доступного конфигурационного пространства и предоставляет более оптимальный метод доступа. Конфигурационная модель PCI Express поддерживает два механизма доступа в конфигурационное пространство:

- PCI-совместимый конфигурационный механизм: 100%-я поддержка двоичной совместимости со спецификацией PCI 2.3 или более поздней, поддержка операционных систем и их соответствующей нумерации шин и конфигурационного ПО.
- Расширенный механизм конфигурирования PCI Express: механизм обеспечивает увеличение размера доступного конфигурационного пространства и оптимизацию механизмов доступа.

Каждый канал PCI Express отображается через структуру моста PCI-to-PCI и имеет связанную с ним логическую шину PCI. Канал PCI Express представлен через структуру моста PCI-to-PCI и может быть портом корневого комплекса, вышележащим или нижележащим портом коммутатора. Корневой порт является структурой моста PCI-to-PCI, которая порождает иерархический домен PCI Express от корневого комплекса. Логические устройства отображаются в конфигурационное пространство таким образом, что каждому будет соответствовать индивидуальный номер устройства [7].

Конфигурационная топология

Для поддержки совместимости с механизмами доступа конфигурационного ПО шины PCI, все элементы PCI Express имеют отображение, совместимое с конфигурационным пространством PCI. Каждый канал PCI Express, порожденный из логического моста PCI-to-PCI, отображается в конфигурационное пространство как вторичная шина этого моста. *Корневой порт* — это структура моста PCI-to-PCI, которая порождает канал PCI Express из корневого комплекса PCI Express (рис. 12.6).

Коммутатор PCI Express представлен несколькими структурами моста PCI-to-PCI, соединяющими каналы PCI Express с внутренней логикой шины PCI. *Upstream-порт коммутатора* — это мост PCI-to-PCI; вторичная шина этого моста представляет внутреннюю маршрутизирующую логику коммутатора. *Downstream-порты коммутатора* являются мостами PCI-to-PCI, соединяющими внутренние шины с шинами, представляющими собой downstream-каналы от коммутатора PCI Express (рис. 12.7).

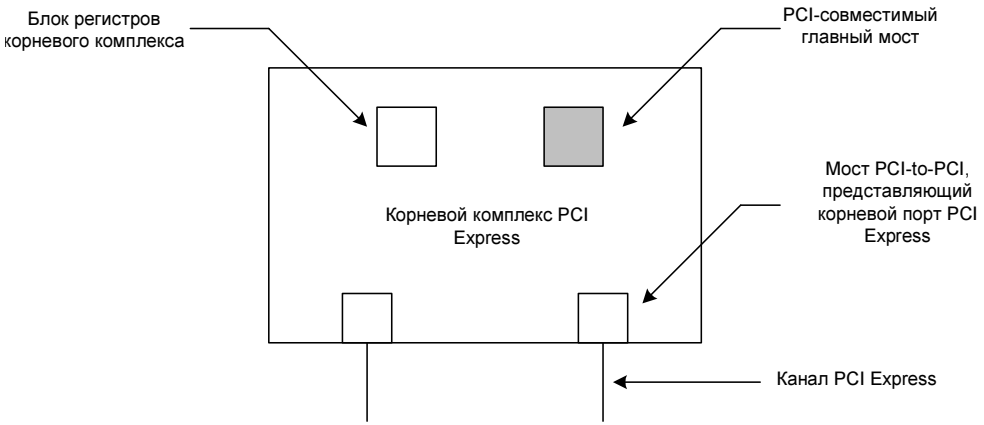


Рис. 12.6. Отображение устройства корневого комплекса

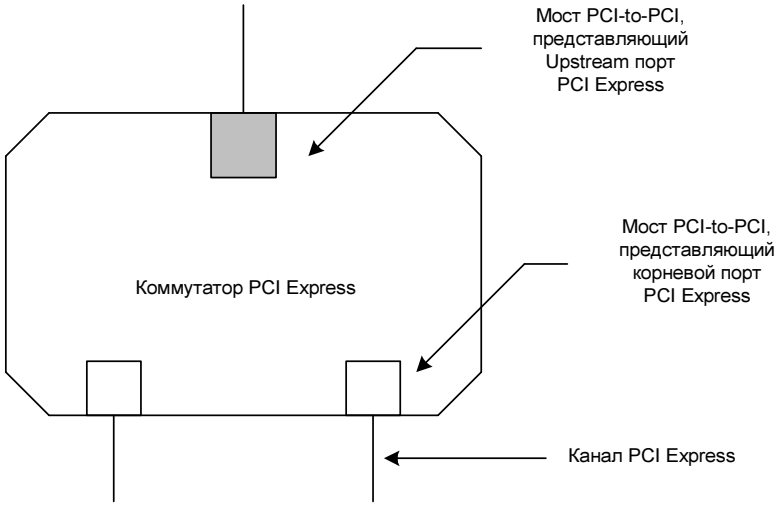


Рис. 12.7. Отображение устройства коммутатора PCI Express [1]

Оконечное устройство PCI Express отображается в конфигурационное пространство как отдельное логическое устройство (устройство 0) с одной или больше логическими функциями [7].

[1] Будущие коммутаторы PCI Express могут быть реализованы как отдельные компоненты Switch Device (без мостов PCI-to-PCI), которые не будут ограничены требованиями обратной совместимости, диктуемые существующим ПО стандарта PCI.

Конфигурационные механизмы PCI Express

PCI Express расширяет конфигурационное пространство до 4096 байт для каждой функции устройства, в то время как спецификация PCI 2.3 обеспечивала только 256 байт. Конфигурационное пространство PCI Express состоит из двух областей, что показано на рис. 12.8. Область, совместимая с PCI 2.3, располагается в первых 256 байтах конфигурационного пространства логического устройства, расширенную область PCI Express занимает оставшееся конфигурационное пространство. Доступ в область, совместимую со спецификацией PCI 2.3, может быть осуществлен через механизм PCI 2.3 или механизм доступа в расширенное пространство PCI Express. Оба механизма производят одинаковые изменения, но не допускается одновременное использование программным обеспечением обоих методов доступа в конфигурационные регистры устройства. Доступ в дополнительную область PCI Express может быть осуществлен только через механизм доступа PCI Express[2] [7].

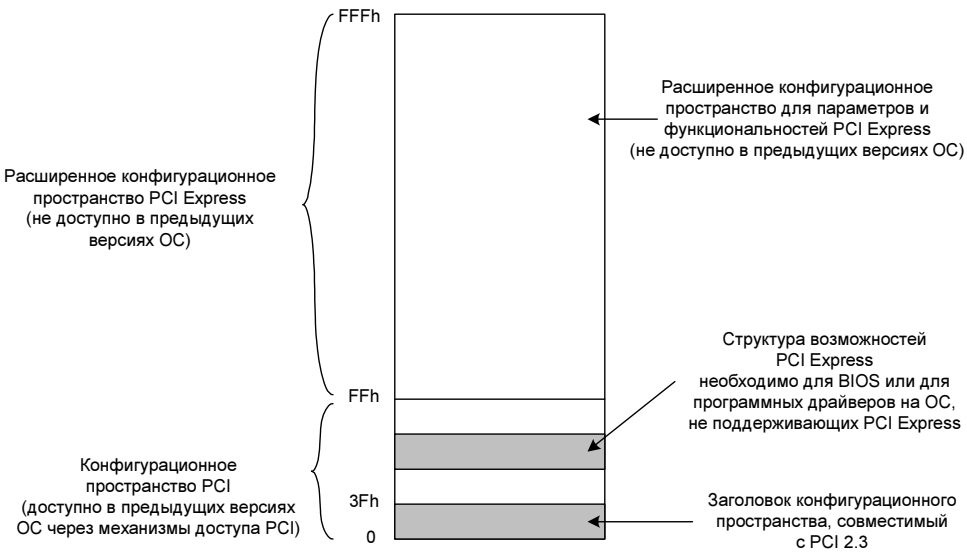


Рис. 12.8. Структура конфигурационного пространства PCI Express

PCI-совместимый механизм доступа

Механизм доступа в конфигурационное пространство PCI Express, совместимый с PCI, поддерживает программную модель конфигурационного про-

[2] Доступы, выполняющиеся только в область PCI Express и использующие механизм доступа PCI Express, могут чередоваться с доступами, использующими механизм PCI.

странства PCI в соответствии со спецификацией PCI 2.3. В соответствии с этой моделью системы, которые содержат интерфейс PCI Express, должны поддерживать нумерацию шин и конфигурационное программное обеспечение PCI.

Как и устройства PCI, устройства PCI Express должны обеспечивать пространство конфигурационных регистров для программно-управляемой инициализации и конфигурирования. За некоторым исключением, регистры заголовка PCI Express организованы в соответствии со спецификацией PCI 2.3.

Совместимый с PCI 2.3 механизм доступа использует такой же формат запроса, что и механизм доступа PCI Express. Для PCI-совместимого конфигурационного запроса все поля расширенного регистра адреса должны быть равны нулю.

Расширенный конфигурационный механизм доступа PCI Express

Расширенный конфигурационный механизм доступа использует отображение в плоское адресное пространство памяти для доступа к конфигурационным регистрам устройства. При такой реализации адрес в памяти определяет конфигурационный регистр, в который осуществляется доступ, а данные в памяти — содержимое адресуемого регистра. Отображение из адресов памяти в адрес конфигурационного пространства PCI Express определено в табл. 12.1. Базовый адрес AD[63:28] назначается в зависимости от реализации и сообщается операционной системе системным ПО.

Таблица 12.1. *Отображение конфигурационных адресов*

Адрес в памяти	Конфигурационное пространство PCI Express
A[27:20]	Шина[7:0]
A[19:15]	Устройство [4:0]
A[14:12]	Функция [2:0]
A[11:8]	Расширенный регистр [3:0]
A[7:0]	Регистр [7:0]

Требования к главному мосту

Главный мост (Host Bridge) PCI Express должен транслировать отображаемые в память доступы в конфигурационное пространство PCI Express, передаваемые от главного процессора в конфигурационную транзакцию PCI Express. Код класса главного моста PCI зарезервирован для обратной совместимости;

конфигурационное пространство главного моста непрозрачно для стандартного ПО PCI Express и может быть реализовано специфически для каждой конкретной реализации, которая совместима с конфигурационным пространством типа 0 главного моста PCI [7].

Требования к устройству PCI Express

Устройства должны поддерживать дополнительные 4 бита для дешифрации доступа к конфигурационному регистру, т. е. они должны дешифрировать поле **Extended Register Address**, биты [3:0] заголовка конфигурационного запроса.

Блок регистров корневого комплекса

Каждый корневой порт связан с блоком отображенных в память регистров размером 4096 байт, который называется как *блок регистров корневого комплекса* или *Root Complex Register Block (RCRB)*. Эти регистры используются аналогично конфигурационному пространству и могут включать расширенные возможности PCI Express и другие специфические регистры, которые применяются к корневому комплексу.

Системное ПО низкого уровня передает операционной системе базовый адрес блоком RCRB для каждого корневого порта. Несколько корневых портов могут быть связаны с одним RCRB. Отображенные в память регистры RCRB не должны размещаться в одинаковом адресном пространстве, как отображенное в память конфигурационное пространство.

Типы конфигурационных регистров

Поля конфигурационных регистров имеют один из атрибутов, описанных в табл. 12.2.

Таблица 12.2. *Типы регистров и полей*

Атрибут регистра	Описание
RO	Read-only. Регистр доступен только для чтения: биты регистра доступны только для чтения и не могут быть изменены ПО
RW	Read-Write. Регистр доступен для чтения/записи: биты регистра доступны для чтения/записи и могут быть изменены или очищены ПО
RW1C	Read-only status, Write-1-to-clear status.. Регистр статуса, доступен только для чтения. Биты регистра при чтении возвращают статус. Все биты регистра могут быть очищены путем записи "1". Запись "0" в биты RW1C не имеет эффекта

Таблица 12.2 *(окончание)*

Атрибут регистра	Описание
ROS	Sticky bit — Read-only. Биты регистра доступны только для чтения и не могут быть изменены ПО. Биты не очищаются после сброса и могут быть сброшены только методом сброса "Power Good Reset". Устройства, которые потребляют питание со входа AUX, не могут сбрасывать эти биты методом "Power Good Reset", пока подано AUX питание
RWS	Sticky bit — Read-Write. Биты регистра доступны для чтения/записи и могут быть изменены ПО. Биты не сбрасываются после сброса и могут быть сброшены только методом сброса "Power Good Reset". Устройства, которые потребляют питание со входа AUX, не могут сбрасывать эти биты методом "Power Good Reset", пока подано питание через AUX
RW1CS	Sticky bit - Read-only status, Write-1-to-clear status. Регистр статуса, доступен только для чтения и не очищается после сброса. Биты регистра при чтении возвращают статус. Совокупность битов, указывающая статус события, может быть очищена путем записи "1". Запись значения "0" в RW1CS не имеет никакого эффекта. Биты не очищаются после сброса и могут быть сброшены только методом сброса "Power Good Reset". Устройства, которые потребляют питание со входа AUX, не могут сбрасывать эти биты методом "Power Good Reset", пока через AUX подается питание
Hwlnit	Hardware Initialized: биты регистра инициализируются программным или аппаратным методом, таким как "связывающий вывод" или через последовательный EEPROM. После инициализации биты доступны только для чтения и могут быть сброшены только методом "Power Good Reset"
RsvdP	(Reserved and Preserved). Зарезервировано для будущих RW-реализаций; при записи в эти биты ПО должно сохранять считанные значения
RsvdZ	(Reserved and Zero). Зарезервировано для будущих RW1C-реализаций; для записи в эти биты ПО должно использовать "0"

PCI-совместимые конфигурационные регистры

Первые 256 байт конфигурационного пространства PCI Express образуют область, совместимую с PCI 2.3. Эта область полностью соответствует конфигурационному пространству PCI 2.3 устройства или функции. Существующие PCI-устройства могут быть также доступны через расширенный механизм доступа PCI Express без дополнительной модификации аппаратной части устройства или программного обеспечения драйвера устройства. Далее описан установленный формат и поведение регистров, совместимых с PCI 2.3 для отображения между PCI 2.3 и PCI Express.

Все регистры и поля, которые не описаны в данном разделе, предполагаются неизмененными по отношению к их определению в спецификации PCI 2.3 [7].

Общее конфигурационное пространство типа 0/1

На рис. 12.9 представлено расположение общих полей регистров конфигурационных заголовков типа 0 и типа 1 стандарта PCI 2.3 для устройств PCI Express.

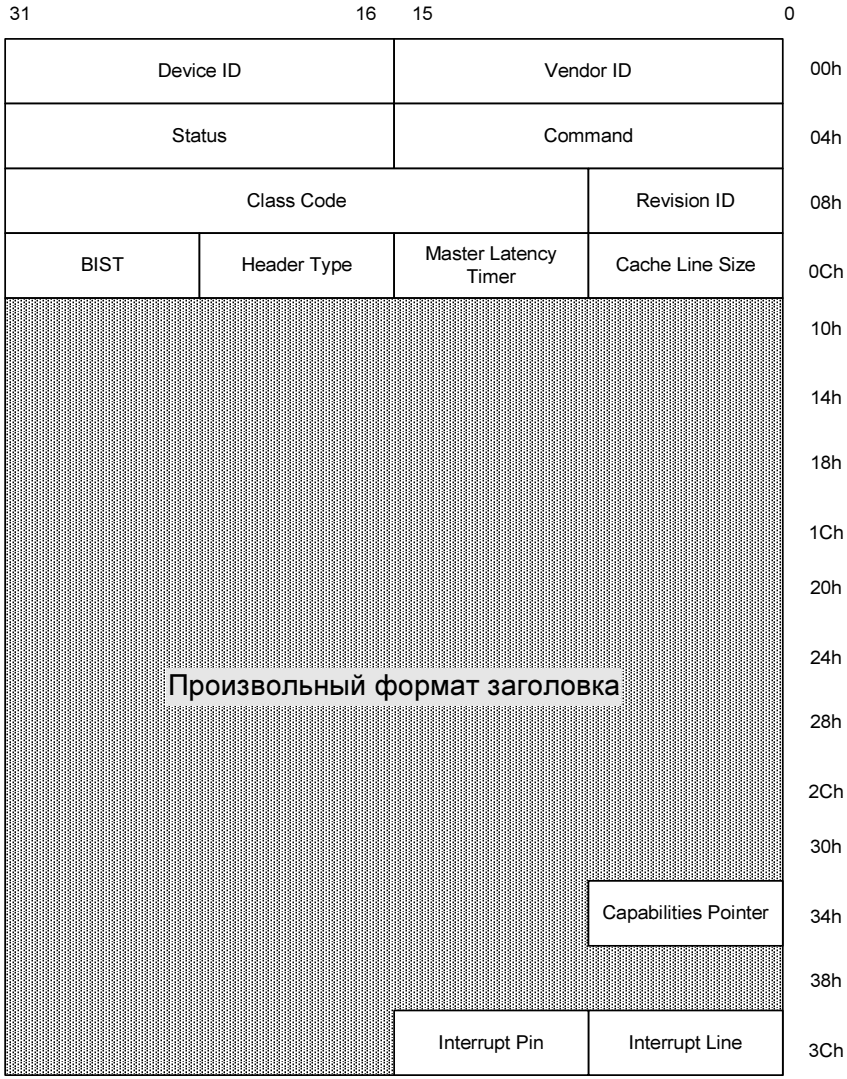


Рис. 12.9. Общий формат заголовка конфигурационного пространства

Эти регистры определены для обоих типов — для типа 0 и типа 1 заголовков конфигурационного пространства. Далее описана специфическая интерпретация этих регистров для PCI Express.

Регистр Command (смещение 04h)

В табл. 12.3 установлено отображение регистра команд конфигурационного пространства из спецификации PCI 2.3 для спецификации PCI Express.

Таблица 12.3. *Регистр Command*

Бит	Описание регистра	Аттр.
02	**Bus Master Enable** — управляет способностью агента PCI Express генерировать запросы чтения/записи в память и в пространство ввода-вывода (ПВВ). Отключение этого бита запрещает агенту PCI Express генерировать любые запросы чтения/записи в память или ПВВ. Сообщения о прерывании MSI являются внутриполосными записями в память, отключение бита **Bus Master Enable** также отключает сообщения о прерывании MSI. По умолчанию значение данного бита равно "0"	RW
03	**Special Cycle Enable** — не применяется по отношению к PCI Express. Бит должен быть аппаратно приведен к "0"	RO
04	**Memory Write and Invalidate** — не применяется по отношению к PCI Express. Бит должен быть аппаратно сброшен в "0"	RO
05	**VGA Palette Snoop** — не применяется по отношению к PCI Express. Бит должен быть аппаратно сброшен в "0"	RO
06	**Parity Error Enable** — по умолчанию значение данного бита равно "0"	RW
07	**IDSEL Stepping / Wait Cycle Control** — не применяется по отношению к PCI Express. Бит должен быть аппаратно сброшен в "0"	RO
08	**SERR# Enable** — будучи установленным, этот бит сообщает о фатальных и не критических ошибках корневому комплексу. Специфические биты регистра ошибки PCI Express имеют приоритет перед этим битом. По умолчанию значение данного бита равно "0"	RW
09	**Fast Back-to-Back Transactions Enable** — не применяется по отношению к PCI Express. Бит должен быть аппаратно сброшен в "0"	RO
10	**Interrupt Disable** — управляет способностью устройства PCI Express генерировать сообщения о прерываниях INTx. Когда этот бит установлен, устройства не могут генерировать сообщения о прерываниях INTx. Любые ранее выставленные эмуляции прерываний INTx должны быть сняты при установке данного бита. По умолчанию значение данного бита равно "0"	RW

Регистр Status (смещение 06h)

В табл. 12.4 представлено отображение регистра статуса конфигурационного пространства из спецификации PCI 2.3 для спецификации PCI Express.

Таблица 12.4. *Регистр Status*

Бит	Описание регистра	Аттр.
03	**Interrupt Status** — указывает, что сообщение о прерывании INTx к устройству задержано внутренне. По умолчанию значение данного бита равно "0"	RO
04	**Capabilities List** — указывает на наличие пункта списка расширенных возможностей. Поскольку все устройства PCI Express должны реализовать структуру возможностей PCI Express, то этот бит должен быть установлен в "1"	RO
05	**66 MHz Capable** — не применяется по отношению к PCI Express. Бит должен быть аппаратно сброшен в "0"	RO
07	**Fast Back-to-Back Transactions Capable** — не применяется по отношению к PCI Express. Бит должен быть аппаратно сброшен в "0"	RO
08	**Master Data Parity Error** — бит устанавливается запросчиком (основная сторона для устройства с заголовком конфигурационного пространства типа 1), если установлен его бит **Parity Error Enable** регистра Command и произошло одно из двух событий: • запросчик принял выполнение, отмеченное как искаженное; • запросчик исказил запрос записи, если сброшен бит **Parity Error Enable** регистра Command, то этот бит уже не может быть изменен. По умолчанию значение данного бита равно "0"	RW1C
10:09	**DEVSEL Timing** — не применяется по отношению к PCI Express. Бит должен быть аппаратно сброшен в "0"	RO
11	**Signaled Target Abort** — бит устанавливается, когда устройство (основная сторона устройства с заголовком конфигурационного пространства типа 1 для запросов, выполненных самим устройством) выполняет запрос, используя статус Completer Abort Completion. По умолчанию значение данного бита равно "0"	RW1C
12	**Received Target Abort** — бит устанавливается, когда запросчик (основная сторона устройства с заголовком конфигурационного пространства типа 1 для запросов, инициированных самим устройством) принимает выполнение со статусом Completer Abort Completion. По умолчанию значение данного бита равно "0"	RW1C

Таблица 12.4 *(окончание)*

Бит	Описание регистра	Аттр.
13	**Received Master Abort** — бит устанавливается, когда запросчик (основная сторона устройства с заголовком конфигурационного пространства типа 1 для запросов, инициированных самим устройством) принимает выполнение со статусом Unsupported Request Completion. По умолчанию значение данного бита равно "0"	RW1C
14	**Signaled System Error** — бит устанавливается, когда устройство посылает сообщение "ERR_FATAL" или "ERR_NONFATAL". По умолчанию значение данного бита равно "0"	RW1C
15	**Detected Parity Error** — бит устанавливается устройством (основная сторона устройства с заголовком конфигурационного пространства типа 1) каждый раз, когда оно принимает искаженный пакет TLP (Transaction Layer Packet), невзирая на состояние бита **Parity Error Enable**. По умолчанию значение данного бита равно "0"	RW1C

Регистр Cache Line Size (смещение 0Ch)

Значение регистра размера кэш-линии устанавливается системным ПО низкого уровня и операционной системой, в него записывается размер системной кэш-линии. Существующее ПО PCI 2.3 может не всегда корректно программировать этот регистр, особенно для устройств Hot-Plug. Этот регистр реализуется устройствами PCI Express как доступный для чтения/записи в целях обратной совместимости, но не влияет на какие-либо функциональные возможности устройств PCI Express.

Регистр Master Latency Timer (смещение 0Dh)

Этот регистр также называется как "основной таймер задержки" для устройств с заголовком конфигурационного пространства типа 1. Основной таймер задержки не применяется по отношению к PCI Express. Данный регистр должен быть аппаратно сброшен в "0".

Регистр Interrupt Line (смещение 3Ch)

Как в PCI 2.3, регистр линии прерывания Interrupt Line сообщает информацию о маршруте линии прерывания. Этот регистр доступен для чтения/записи и должен быть реализован любым устройством (или функцией устройства), которая использует вывод прерывания (см. последующее описание). Значение этого регистра программируется системным ПО и является специфическим для системной архитектуры. Само устройство не использует

данное значение; это значение используется драйверами устройства и операционными системами.

Регистр Interrupt Pin (смещение 3Dh)

Регистр доступен только для чтения и идентифицирует существующие и устаревшие сообщения о прерывании, используемые устройством или функцией устройства; разрешенными значениями являются 1, 2, 3 и 4. Эти значения отображаются на сообщения о прерывании для INTA, INTB, INTC и INTD соответственно; значение "0" указывает, что устройство не использует сообщения о прерывании.

Регистры ошибок

Биты регистра контроля/статуса ошибок в регистрах Command и Status управляют PCI-совместимым сообщением об ошибках для устройств PCI и PCI Express. В дополнение к PCI-совместимому контролю за ошибками, сообщение об ошибках PCI Express может управляться отдельно от устройств PCI через структуру функциональностей PCI Express. PCI-совместимые поля регистра статуса и контроля за ошибками не имеют значения для сообщений об ошибках PCI Express, определенных через структуру функциональностей PCI Express. Устройства PCI Express могут также дополнительно реализовывать расширенные сообщения об ошибках.

Заголовок конфигурационного пространства типа 0

На рис. 12.10 представлено размещение полей регистров заголовка типа 0 конфигурационного пространства PCI 2.3 для устройств PCI Express.

Заголовок конфигурационного пространства типа 1

На рис. 12.11 показано расположение полей регистров заголовка типа 1 конфигурационного пространства PCI 2.3 для устройств PCI Express.

В данном разделе описана интерпретация специфических PCI Express регистров для определенных регистров заголовка типа 1 конфигурационного пространства PCI 2.3.

Регистр Secondary Latency Timer (смещение 1Bh)

Этот регистр не применяется по отношению к PCI Express и должен быть аппаратно сброшен в "0".

Регистр Secondary Status (смещение 1Eh)

Табл. 12.5 устанавливает отображение для регистра конфигурационного пространства Secondary Status из спецификации PCI 2.3 для спецификации PCI Express.

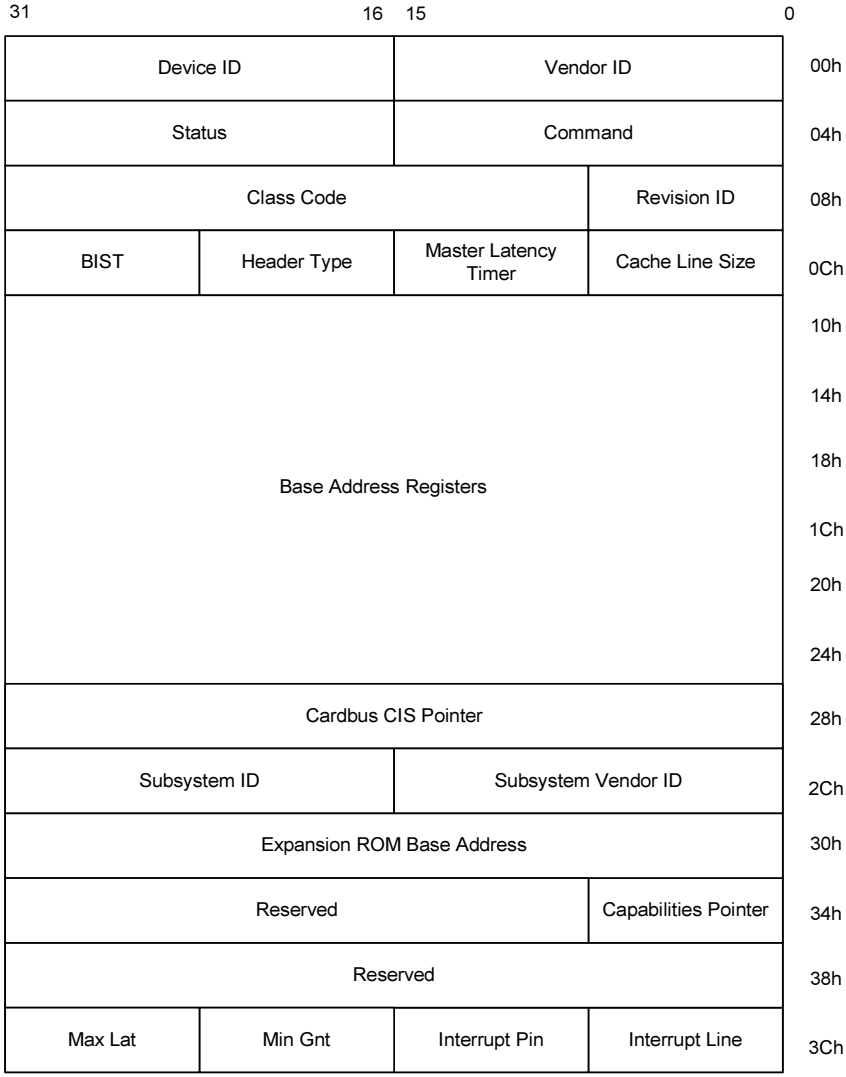


Рис. 12.10. Формат заголовка конфигурационного пространства типа 0

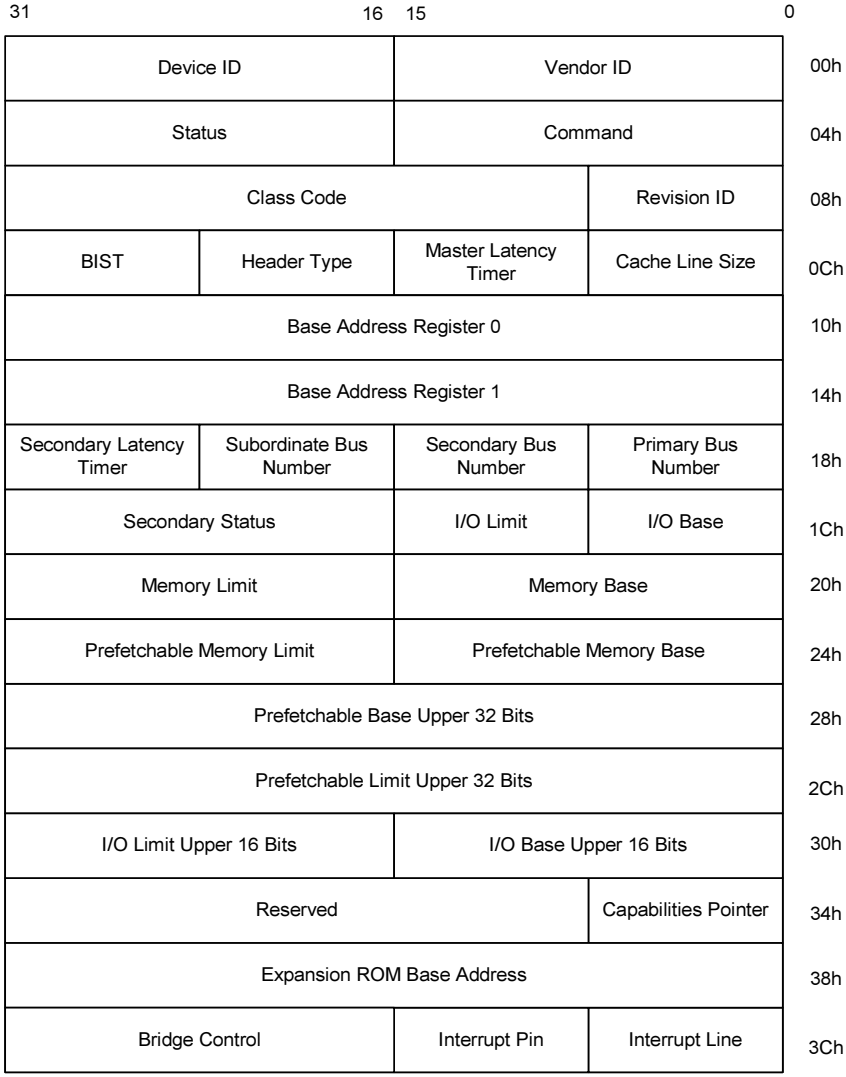


Рис. 12.11. Формат заголовка конфигурационного пространства типа 1

Таблица 12.5. *Регистр Secondary Status*

Бит	Описание регистра	Аттр.
05	**66 MHz Capable** – не применяется к PCI Express и должен быть аппаратно сброшен в "0"	RO
07	**Fast Back-to-Back Transactions Capable** — не применяется к PCI Express и должен быть аппаратно сброшен в "0"	RO
08	**Master Data Parity Error** — бит устанавливается запросчиком вторичной стороны при условии, что установлен бит **Parity Error Response Enable** в регистре Bridge Control и произошло одно из двух событий: • запросчик принял выполнение, помеченное как искаженное; • запросчик исказил запрос записи. Если очищен бит **Parity Error Response Enable** в регистре Bridge Control, то он не может быть изменен. По умолчанию значение данного бита равно "0"	RW1C
10:09	**DEVSEL Timing** — поле не применяется к PCI Express и должно быть аппаратно сброшено в "0"	RO
11	**Signaled Target Abort** — бит устанавливается, когда вторичная сторона для устройства с заголовком конфигурационного пространства типа 1 (для запросов, выполненных самим устройством) выполняет запрос, используя статус "Completer Abort Completion". По умолчанию значение данного бита равно "0"	RW1C
12	**Received Target Abort** — бит устанавливается, когда вторичная сторона для устройства с типом 1 заголовка конфигурационного пространства (для запросов, инициированных самим устройством) принимает выполнение со статусом "Completer Abort Completion". По умолчанию значение данного бита равно "0"	RW1C
13	**Received Master Abort** — бит устанавливается, когда вторичная сторона для устройства с заголовком конфигурационного пространства типа 1 (для запросов, инициированных самим устройством) принимает выполнение со статусом "Unsupported Request Completion". По умолчанию значение данного бита равно "0"	RW1C
14	**Received System Error** — бит устанавливается, когда устройство посылает сообщения "ERR_FATAL" или "ERR_NONFATAL". По умолчанию значение данного бита равно "0"	RW1C
15	**Detected Parity Error** — бит устанавливается вторичной стороной для устройства с заголовком конфигурационного пространства типа 1 всякий раз, когда оно принимает искаженный пакет TLP, независимо от состояния бита **Parity Error Response Enable** в регистре Bridge Control. По умолчанию значение данного бита равно "0"	RW1C

Регистр Bridge Control (смещение 3Eh)

В табл. 12.6 представлено отображение для регистра Bridge Control из спецификации PCI 2.3 для спецификации PCI Express.

Таблица 12.6. *Регистр Bridge Control*

Бит	Описание регистра	Аттр.
00	**Parity Error Response Enable** — бит управляет откликом на искаженные пакеты TLP (Transaction Layer Packet). По умолчанию значение данного бита равно "0"	RW
01	**SERR# Enable** — бит управляет направлением сообщений "ERR_COR", "ERR_NONFATAL" и "ERR_FATAL" от вторичной к первичной стороне. По умолчанию значение данного бита равно "0"	RW
05	**Master Abort Mode** — не применяется в PCI Express и должен быть аппаратно сброшен в "0"	RO
06	**Secondary Bus Reset** — установка этого бита приводит к "теплому" сбросу на соответствующем порту PCI Express и подчиненном порту иерархического домена PCI Express. По умолчанию значение данного бита равно "0"	RW
07	**Fast Back-to-Back Transactions Enable** — не применяется к PCI Express и должен быть аппаратно сброшен в "0"	RO
08	**Primary Discard Timer** — не применяется к PCI Express и должен быть аппаратно сброшен в "0"	RO
09	**Secondary Discard Timer** — не применяется к PCI Express и должен быть аппаратно сброшен в "0"	RO
10	**Discard Timer Status** — не применяется к PCI Express и должен быть аппаратно сброшен в "0"	RO
11	**Discard Timer SERR Enable** — не применяется к PCI Express и должен быть аппаратно сброшен в "0"	RO

Управление питанием

В данном разделе описаны функции и протоколы управления питанием PCI Express так называемого механизма *PCI Express-PM*.

PCI Express-PM предоставляет следующие сервисы:

- механизм для идентификации функциональных возможностей управления питанием данной функции;

- способность переводить функцию в определенное состояние энергопотребления;
- сообщение текущего состояния энергопотребления функции;
- возможность пробуждения системы по определенному событию.

PCI Express-PM совместим со спецификациями "PCI Bus Power Management Interface" версии 1.1 (PCI-PM) и "Advanced Configuration and Power Interface" версии 2.0 (ACPI). В данном разделе также определены функциональные возможности собственного механизма управления питанием PCI Express. Он предоставляет дополнительные возможности управления питанием по сравнению со спецификацией "PCI Power Management Interface".

PCI Express-PM определяет состояния, в которые может переходить физический канал PCI Express в ответ на программно-управляемые переходы в D-состояния или при активизации механизма Active State Link PM. Состояния каналов PCI Express "не видны" напрямую для существующего ПО драйвера шины, но информация может быть получена из состояния управления питанием компонентов, размещенных на этих каналах. Определены состояния канала L0, L0s, L1, L2 и L3. Уменьшение потребления питания направлено при переходе от состояния L0 к L3.

Компоненты PCI Express могут пробуждать систему из любого поддерживаемого состояния энергопотребления через запрос события управления питанием (PME — Power Management Event). Системы PCI Express могут обеспечивать дополнительный источник питания (Vaux), необходимый для операций PME из состояния системы "откл.". В этом отношении механизм PCI Express-PM более расширен по сравнению с его прототипом PCI-PM, это выражается в поддержке PCI Express "сообщений" PME, включающие географическое размещение (Requestor ID) внутри иерархии запрашивающего агента. Данные сообщения PME являются внутриполосными пакетами TLP, направленными из запрашивающего устройства в корневой комплекс.

Другое отличие механизма PCI Express-PM от PME состоит в его разделении двух следующих событий, связанных с PME:

- пробуждение иерархии ввода-вывода (т. е. запуск синхронизирующих и основных шин питания, соединенных с компонентами PCI Express);
- посылка сообщения PME (вектора) корневому комплексу.

Самоуправляющийся аппаратный механизм активного состояния (Active State Link PM) допускает потребление питания, даже когда присоединенные компоненты находятся в состоянии D0. По истечении некоторого периода ожидания (простоя) канала механизм "Active State Link PM" приводит в действие протокол физического уровня, который переводит свободный канал в состояние с низким потреблением питания. Переход из состояния низкого потребления в

полнофункциональное состояние L0 запускается появлением трафика на обеих сторонах канала. Конечные точки инициируют вход в состояние низкого потребления питания канала. Данная функция может быть отключена программным обеспечением.

В терминологии PCI Express термин Upstream-компонент или Upstream-устройство обозначает компонент PCI Express, находящийся на конце канала PCI Express и являющийся закрывающим устройством по отношению к корню иерархического дерева PCI Express. Термин *Downstream-компонент* или *Downstream-устройство* обозначает компонент PCI Express, находящийся на конце канала, который иерархически отдален от корня иерархического дерева PCI Express.

Все компоненты PCI Express, за исключением корневого комплекса, должны выполнять минимум требований, определенных программным обеспечением PCI-PM, совместимым с функциями PCI Express-PM. Корневые комплексы должны участвовать в управлении питанием канала протоколов DLLP, инициированных downstream-устройством, когда все функции downstream-компонента входят в состояние с низким потреблением, совместимое с программным обеспечением PCI-PM.

Функциональности Active State Link PM являются обязательными (вход в состояние L0s как минимум) для всех компонентов, включая корневые комплексы, и конфигурируются раздельно через собственные конфигурационные механизмы PCI Express [7].

Состояния потребления питания канала

Стандарт PCI Express определяет состояния энергопотребления канала вместо состояний энергопотребления шины, которое было определено в спецификации PCI-PM. Информация о состоянии канала не доступна для существующего ПО, совместимого с PCI-PM, и может быть получена либо через D-состояния соответствующего компонента, соединенного с каналом, либо через протоколы управления питанием "Active State". Физический уровень PCI Express может определить дополнительные промежуточные состояния.

PCI Express-PM устанавливает следующие состояния энергопотребления канала.

- **L0** — активное состояние. Разрешены все транзакции PCI Express и другие операции. Поддержка состояния L0 требуется для управления питанием Active State Link и для PCI-PM-совместимой модели.
- **L0s** — состояние с малой задержкой возобновления работы, экономичное потребление типа "Standby". Поддержка состояния L0s обязательна для модели Active State Link и не применима к модели совместимой с PCI-PM.

Все основные источники питания, генераторы тактовых импульсов компонентов и внутрикомпонентные ФАПЧ должны находиться в активном состоянии при нахождении системы в состоянии L0s. Передача информации пакетами TLP и DLLP через канал, который находится в состоянии L0s, запрещена. Состояние L0s используется исключительно для активного состояния. Физический уровень PCI Express обеспечивает механизм для быстрых переходов из этого состояния в состояние L0. Когда общие (распределенные) генераторы синхроимпульсов используются на обеих сторонах данного канала, время перехода из состояния L0s в L0 обычно меньше, чем 100 нс.

- **L1** — состояние с большой задержкой, низким потреблением питания типа "Standby". Поддержка состояния L1 обязательна для модели совместимой с PCI-PM и необязательна для модели Active State Link. Все содержащиеся на платформе основные источники питания и компонентные генераторы синхроимпульсов должны оставаться в активном состоянии все время нахождения системы в состоянии L1. Внутренние ФАПЧ downstream-компонента могут быть отключены при нахождении в состоянии L1, обеспечивая большую экономию энергии за счет увеличения выходного времени задержки[3]. Переход в состояние L1 осуществляется, когда все функции downstream-компонента данного канала PCI Express запрограммированы в D-состояние, отличное от состояния D0, либо, если downstream-компонент запросил переход в состояние L1 (Active State Link PM) и получил положительный ответ. Выход из состояния L1 происходит при инициировании транзакции со стороны upstream-устройства, направленной в downstream-компонент, или если downstream-компонент хочет инициировать транзакцию по направлению к upstream-устройству. Переход из состояния L1 в L0 занимает несколько микросекунд. Передача информации пакетами TLP и DLLP через канал при нахождении его в состоянии L1 запрещена.
- **L2/L3 Ready** — состояние не предназначено специально для переходов D-состояний PCI-PM или для управления питанием Active State Link. Канал переходит в состояние L2/L3 Ready, когда платформа готова перейти в состояние покоя (Sleep). После выполнения протокола перехода в состояние L2/L3 Ready канал готов к переходу в состояние L2 или L3, но не перейдет ни в одно из них до тех пор, пока не будет снято основное питание. В зависимости от реализации, для конкретной платформы после выключения основного питания канал может переходить в состояние L2 (в случае, если присутствует источник Vaux) или в состояние отключения "off" (L3). Процесс

[3] Например, отключение внутреннего ФАПЧ может быть желательно при нахождении в состоянии D3hot, но не желательно для состояний D1 или D2.

перехода в состояние L2/L3 Ready должен начаться после получения уведомления — сообщения "PM_TURN_OFF" (т. е. TLP-пакета "PM_TO_Ack"). Downstream-компонент инициирует переход в состояние L2/L3 Ready путем выдачи DLLP-пакета "PM_Enter_L23" в передающий порт. Передачи пакетов TLP и DLLP через канал при нахождении в состоянии L2/L3 Ready запрещены. Выход из состояния L2/L3 Ready обратно в состояние L0 может быть инициирован только транзакцией, инициированной со стороны upstream и направленной к downstream-компоненту. Механизм такой же, как и при "отпирании" компонента upstream-транзакцией из состояния L1 в L0. Выход из состояния L2/L3 Ready, инициированный со стороны upstream, приведет к тому, что при переходе канала в состояние L2/L3 Ready до выключения основного питания менеджер питания платформы примет решение не входить в состояние Sleep. Переход канала в состояние L2/L3 Ready является одной из заключительных стадий, относящихся к протоколу PCI Express при подготовке к переходу платформы в состояние Sleep, когда снимается основное питание (например, состояние ACPI S3 или S4).

- **L2** — состояние экономичного потребления питания с дополнительного источника. Поддержка состояния L2 является необязательной, и зависит от поддержки платформой источника Vaux. При нахождении в состоянии L2 вход источника питания компонента и вход генератора синхроимпульсов отключены. При нахождении в состоянии L2 вся детектирующая логика PME, логика возобновления канала типа "Beacon", контекст PME и любая другая "оживляющая" логика запитывается от источника Vaux. Передачи пакетов TLP и DLLP через канал, который находится в состоянии L2, запрещены. Выход из состояния L2 завершается путем восстановления основного питания и генератора синхроимпульсов для всех компонентов внутри области менеджера питания, это происходит после полной регулировки и инициализации канала. Как только данный канал выполнит регулировку и инициализацию, то он находится в состоянии L0 и может посылать и принимать пакеты TLP и DLLP.
- **L3** — состояние отключения канала, напряжение питания равно нулю.

На рис. 12.12 приведен граф разрешенных переходов между L-состояниями, которые могут происходить во время функционирования канала.

> ***Примечание***
>
> Дуга, обозначенная отдельно на рис. 12.12, показывает случай, когда на платформе не реализован дополнительный источник Vaux. В этом случае, протокол перехода из состояния L2/L3 Ready приводит к готовности снятия основного питания, и как только оно снимается, канал переходит в состояние L3.

Переходы канала PM из любого L-состояния в любое другое L-состояние должны проходить через состояние L0 во время процесса перехода, исключая

переходы из состояния L2/L3 Ready в состояние L2 или L3. В этом случае, переходы канала из состояния L2/L3 Ready направлены в состояние L2 или L3, когда снято основное питание компонента. (Это следует одновременно с соответствующим переходом компонента из состояния $D3_{hot}$ в $D3_{cold}$, т. е. из состояния пассивного потребления в состояние практически полного отключения.)

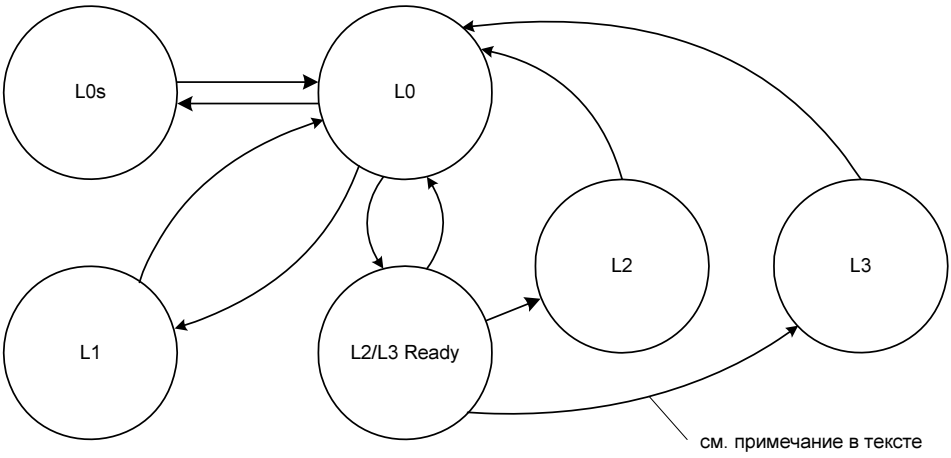


Рис. 12.12. Граф допустимых переходов между состояниями канала

Следующая последовательность, подготавливающая систему к переходу в состояние Sleep, иллюстрирует многошаговый процесс перехода канала из состояния в состояние:

1. Системное ПО направляет все функции downstream-компонента в состояние $D3_{hot}$.
2. Downstream-компонент инициирует переход канала в состояние L1 установленным порядком.
3. Системное ПО заставляет корневой комплекс послать широковещательное сообщение "PM_Turn_Off", подготавливающее к снятию основного питания.
4. Данное сообщение заставляет перейти подчиненный канал обратно в состояние L0, чтобы послать это сообщение, что позволяет downstream-компоненту ответить сообщением "PM_TO_Ack".
5. После посылки сообщения "PM_TO_Ack" downstream-компонент инициирует протокол перехода L2/L3 Ready.

$$L0 \rightarrow L1 \rightarrow L0 \rightarrow L2/L3\ Ready$$

В табл. 12.7 обобщены все L-состояния, условия их использования и поведение платформы PCI Express и компонентов PCI Express для каждого из них.

Значение поля "Yes" указывает, что поддержка обязательна (кроме специально указанных ниже). Значения "On" и "Off" указывают на наличие и отсутствие синхронизации и питания соответственно. "On/Off" указывает на любую реализацию данной опции.

Таблица 12.7. *Обобщение состояний энергопотребления канала PCI Express*

L-состояния	Описание	Использование ПО управления PM	Использование Active State LinkPM	Генератор синхроимпульсов платформы	Основное питание платформы	Внутренний ФАПЧ компонента	Vaux платформы
L0	Полностью активный канал	Yes (D0)	Yes (D0)	On	On	On	On/Off
L0s	Резервное состояние (Standby)	No	Yes[1] (D0)	On	On	On	On/Off
L1	Резервное состояние малого потребления (Standby)	Yes (D1—$D3_{hot}$)	Yes[2] (opt., D0)	On	On	On/Off[3]	On/Off
L2/L3 Ready	Точка готовности к снятию питания	Yes[4]	No	On	On	On/Off	On/Off
L2	Состояние с низким потреблением (Sleep). (Снято основное питание и синхронизация)	Yes[5]	No	Off	Off	Off	On[6]
L3	Откл. Питание снято	—	—	Off	Off	Off	Off

Примечания

1. Задержка выхода из состояния L0s будет большей для конфигурации канала, который характеризуется независимыми входами генераторов синхроимпульсов для компонента, присоединенного с другой стороны данного канала (вместо общего используется распределенный генератор синхросигналов).
2. Переход в состояние L1 может быть запрошен внутри протокола Active State Link PM, однако данная поддержка необязательна.
3. Задержка выхода из состояния L1 будет более большой для компонентов, которые производят отключение их ФАПЧ внутренне при нахождении в этом состоянии.

4. Последовательность входа в состояние L2/L3 инициируется выполнением протокола рукопожатия "PM_Turn_Off" / "PM_TO_Ack". Данное правило не примыкает к переходам из D-состояний или переходам в соответствии с политикой и процедурами Active State Link PM.
5. В зависимости от реализации платформы, Sleep-состояния системы могут использовать состояние L2 или состояние полного отключения (L3). Протокол перехода состояния L2/L3 Ready инициируется downstream-компонентом после приема и подтверждения TLP-сообщения "PM_Turn_Off". Если поддержка состояния L2 необязательна для конфигурации (т. е. для питания Vaux), то поддержка компонентом протокола PCI Express для перехода канала в состояние L2/L3 Ready является обязательной.
6. Состояние L2 отличается от L3 только наличием источника Vaux. После выполнения протокола перехода состояния L2/L3 Ready и перед тем, как будет снято основное питание, канал укажет его готовность к снятию питания [7].

Системная архитектура PCI Express

Данный раздел описывает различные аспекты межкомпонентных связей архитектуры PCI Express в контексте платформы. Он охватывает детали поддержки прерываний, сообщения об ошибках и их протоколирование, виртуальные каналы, изохронную поддержку, Hot Plug и монопольный доступ (блокировку).

Поддержка прерываний

Модель прерываний PCI Express поддерживает два механизма:

- ❒ эмуляцию INTx;
- ❒ поддержку сообщения о прерывании, иначе MSI (Message Signaled Interrupt).

В целях обратной совместимости PCI Express предоставляет механизм эмуляции PCI INTx для сообщения прерываний системному контроллеру прерываний (обычно как часть системной базовой логики). Данный механизм совместим с существующим программным обеспечением PCI, и обеспечивает такой же уровень и тип обслуживания, как соответствующий механизм сообщений о прерывании PCI, при этом он является независимым от особенностей системного контроллера прерываний. Механизм обратной совместимости допускает поддержку загрузочных устройств без необходимости комплекса сервисных стеков конфигурирования/управления прерываниями уровня BIOS. Он виртуализирует физические сигналы прерываний PCI путем использования механизма внутриполосной системы сигналов.

В дополнение к PCI INTx-совместимой эмуляции прерываний, PCI Express поддерживает механизм сообщения о прерывании (MSI). Механизм PCI Express MSI является совместимым с функциональностью MSI, определенной в спецификации PCI версии 2.3.

Модель прерываний PCI Express

PCI Express применяет эволюционный подход от PCI по отношению к поддержке прерываний.

Как требуется для механизмов прерываний PCI/PCI-X, каждое устройство должно использовать различные режимы для функционирования INTx и MSI. Сложно требовать от устройства PCI Express поддержки обеих схем, не отличающихся от используемых для устройств PCI/PCI-X. Данный подход обладает следующими преимуществами и недостатками:

- ❒ совместимостью с существующими программными моделями PCI;
- ❒ непосредственной поддержкой для загрузочных устройств;
- ❒ постепенным прекращением поддержки (Easier End of Life-EOL) механизмов INTx.

Существующая программная модель используется для разделения существующего режима (INTx) и режима функционирования MSI; таким образом, для PCI Express не требуется специальной программной поддержки.

Программная модель PME

С точки зрения программного обеспечения, механизм PME (Power Management Event) ведет себя подобно запускаемому фронтом сигналу прерывания, что отличается от запускаемого уровнем механизма PME, используемого для PCI. Однако это не влияет на совместимость программного обеспечения операционной системы ввиду того, что PME сообщается операционной системе абстрактно через ACPI BIOS. Текущие ACPI-совместимые операционные системы поддерживают оба режима для PME-запускаемый фронтом и запускаемый уровнем сигнал прерывания.

Чтобы сообщить о PME, устройство PCI Express генерирует сообщение PME на канале PCI Express. Системная логика управления питанием, которая обычно является частью основной логики (чипсета), принимает это сообщение PME и выставляет признак события (ACPI General Purpose Event — GPE), соответствующий сообщению PME. Код ACPI ASL может использовать ID запросчика в "PM_PME" для информирования операционной системы о том, какое устройство вызвало пробуждение [7].

Архитектура PCI Express гарантирует достоверную доставку сообщения PME, т. к. сообщения PCI Express передаются посредством пакетов TLP [7].

Маршрут PME между иерархиями PCI Express и PCI

PME-совместимые PCI-устройства выставляют сигнал PME#, чтобы сообщить о событии управления питанием. Физически сигнал PME# от устройства

PCI может быть либо конвертирован во внутриполосное сообщение PME PCI Express мостом PCI Express-to-PCI, либо перенаправлен непосредственно на вывод GPE главной логики чипсета. Доставка сообщения PME от устройств PCI является спецификой реализации, как в уже существующих PCI-системах. Во время преобразования из PME PCI запускаемого уровнем в запускаемое фронтом сообщение PCI Express PME нельзя потерять ни одного сообщения PME от устройств PCI. Такой преобразующий механизм может также приводить к тому, что будет сгенерировано ложное сообщение PME [7].

Сообщение об ошибках и протоколирование

В стандарте PCI Express не проводится различий между ошибками, которые должны быть проверены, и ошибками, которые проверять необязательно. Каждая такая ошибка ассоциируется либо с портом, либо с определенным устройством (или функцией в многофункциональном устройстве), и эта ассоциация передается дальше с описанием ошибки. Данный раздел описывает, как сообщаются и классифицируются ошибки.

Введение

Рассмотрение включает ошибки, которые происходят на самом интерфейсе PCI Express, и ошибки, которые происходят из-за транзакций, инициированных на PCI Express. Ошибки, происходящие внутри компонента и не связанные с отдельной транзакцией PCI Express, не описываются. Данный класс ошибок обрабатывается через собственный метод с использованием специфических для устройства прерываний для сообщения об ошибке.

PCI Express определяет две парадигмы сообщения об ошибках: основная функциональность и расширенная функциональность "Advanced Error Reporting". Основная функциональность сообщения об ошибках реализовывается во всех PCI Express устройствах и определяет минимум требований сообщения об ошибках. Функциональность "Advanced Error Reporting" определена для большинства сообщений о критических ошибках и реализовывается через определенную структуру функциональностей PCI Express.

Все устройства PCI Express поддерживают существующее, разработанное до введения стандарта PCI Express, ПО для обработки ошибок путем отображения ошибок PCI Express на существующие механизмы сообщения PCI, в дополнение к механизмам PCI Express [7].

Классификация ошибок

Ошибки PCI Express могут быть разделены на два типа:

- некорректируемые (неисправляемые);
- корректируемые (исправляемые) ошибки.

Данная классификация основана на последствиях ошибок, которые приводят к сбою функциональности и снижению производительности. Неисправляемые ошибки могут быть классифицированы далее на:

- ❒ фатальные;
- ❒ не фатальные.

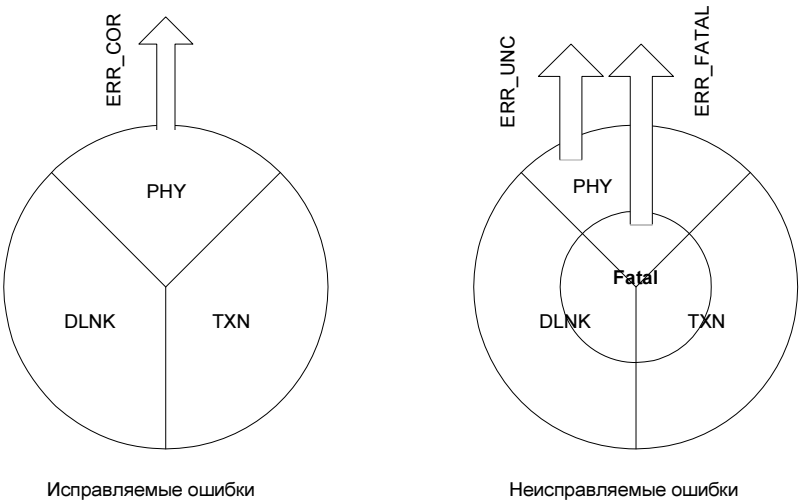


Рис. 12.13. Классификация ошибок

Строгость разделения ошибок на фатальные, некорректируемые и корректируемые позволяет реализовать отображение ошибок для их последующей обработки. Например, платформа может сообщать об исправляемой ошибке через низкоприоритетное ПО, предназначенное для контроля производительности. Такое ПО будет считать частоту исправляемых ошибок и снабжать канал информацией целостности. С другой стороны, разработчик платформы может отображать фатальные ошибки на механизм полного системного сброса. Отображение данных уровней ошибок на различные механизмы обработки не установлено стандартом и определяется разработчиком платформы.

Правила обработки ошибок и PCI-отображения для мостов

Пакет TLP передается либо от одной стороны виртуального PCI-моста к другой, или обрабатывается на входной стороне моста в соответствии с теми же правилами, которые применяются к конечному получателю пакета TLP. Следующие правила охватывают специфические случаи взаимоотношения ошибок PCI Express:

- если запрос не адресован в пространство, отображенное на выходной стороне моста, то он завершается на входной стороне как неподдерживаемый запрос —"Unsupported Request";
- искаженные пакеты TLP перенаправляются в соответствии с теми же правилами, что и нормальные.

При передаче искаженных пакетов TLP:

- принимающая сторона должна установить бит **Detected Parity Error** в регистре Status (Secondary Status);
- передающая сторона должна установить бит **Master Data Parity Error** в регистре Secondary Status, если установлен бит **Parity Error Response Enable** в регистре Bridge Control;
- сообщения "ERR_COR", "ERR_NONFATAL" и "ERR_FATAL" перенаправляются из вторичного интерфейса в основной интерфейс, если установлен бит **SERR# Enable** в регистрах Command и Bridge Control.

Поддержка виртуальных каналов

Механизм *виртуальных каналов*, иначе *VC* (Virtual Channel), является "фундаментом" для поддержки различных сервисов внутри структуры PCI Express. Он разрешает развертывание физически независимых ресурсов, которые в совокупности с маркировкой трафика необходимы для оптимизированной обработки разнородного трафика. Маркировка трафика поддерживается через использование маркеров (меток) *класса транзакций* или, иначе, *TC* (Transaction Class) TLP-уровня. Точная политика для разнородного трафика регулируется путем отображения TC/VC и путем арбитража на уровне VC. Отображение TC/VC зависит от требований применения платформы. Эти требования управляют выбором алгоритма арбитража VC и конфигурируемостью/программируемостью арбитров, что обеспечивает точную настройку политики обслуживания трафика [7].

Далее рассмотрены механизмы виртуальных каналов (VC) с прицелом на будущее. Они адресуют следующие уровни детализации:

- поддерживаемые конфигурации TC/VC;
- правила и алгоритмы арбитража, основанного на VC;
- рассмотрение упорядочивания трафика;
- изохронную поддержку, как определенную пользовательскую модель.

Поддерживаемые конфигурации TC/VC

Виртуальный канал (VC) устанавливается, когда один или несколько классов транзакций (TC) связываются с физическим ресурсом VC, назначенным

идентификатором ID VC. Каждый поддерживаемый класс трафика "Traffic Class" должен быть отображен в один из виртуальных каналов. Основная (базовая) конфигурация PCI Express должна поддерживать пару по умолчанию TC0/VC0, которая является постоянной, т. е. не конфигурируемой. Любая поддержка выше данного уровня является необязательной. Процесс конфигурирования TC/VC управляется системным ПО, использующим программную модель.

Для упрощения взаимодействия при конфигурировании множества виртуальных каналов через канал PCI Express, в стандарте определены ограничения для множества разрешенных (правильных) конфигураций VC. В общем, отображение трафика в виртуальный канал, отличный от TC0/VC0, является задачей системного ПО. Основными конфигурациями TC/VC являются следующие:

- симметричное отображение трафика в виртуальный канал;
- реотображение трафика в виртуальный канал.

Многопортовые компоненты (коммутаторы и корневые комплексы) должны поддерживать независимое отображение TC/VC для каждого порта PCI Express [7].

Механизм "Device Synchronization Stop"

Изменение нумерации шин системным ПО во время функционирования системы может привести к изменению ID запросчика данного устройства (основанного на номерах шин); это может привести к тому, что любые запросы или выполнения для этого устройства еще в процессе их передачи могут быть изменены неверно в соответствии с изменением в ID запросчика. Также желательно гарантировать, чтобы во время извлечения устройства Hot-Plug не происходило никаких выходящих транзакций. Механизм "Device Synchronization Stop" позволяет системному ПО гарантировать, что в процессе передачи по отношению к конкретному оконечному устройству не будет находиться никаких транзакций до выполнения операции ренумерации шин, которая может привести к изменению номера данного устройства (и ID запросчика).

Синхронная остановка для оконечных устройств реализована через механизм "Stop" и связана с битом **Stop** регистра Device Command и битом **Transactions Pending** регистра Device Status .

Системное ПО сообщает устройству требование остановиться путем установки бита **Stop** в регистре Device Command устройства. ПО считает операцию остановки выполненной, если устройство сообщает, что больше нет незавершенных транзакций путем сбрасывания бита статуса **Transactions Pending** в регистре Device Status; при этом устройству не запрещено издавать любой новый запрос после того, как установлен бит **Stop** [7].

Прежде чем сбросить бит **Transaction Pending**, оконечное устройство должно гарантировать, что:

- выполнения небуферизуемых запросов для всех используемых классов трафика были приняты соответствующими запросчиками;
- все запросы, инициированные данным устройством, имеют возвращенные выполнения;
- все буферизованные запросы всех классов трафика были "очищены/сброшены" (т. е. были приняты предназначенными целями) во всех направлениях между оконечным устройством и системой и между одноранговыми оконечными устройствами.

Механизмы очищения/сброса

В самом простом случае, когда оконечное устройство соединяется только с главной памятью, очистка может быть реализована путем направленного чтения памяти. Чтение памяти должно быть выполнено на всех классах трафика, которые использует устройство. Если устройство имеет незавершенные транзакции (включая незавершенные выполнения), тогда оно должно использовать небуферизуемые транзакции типа направленного чтения памяти, адресованное в определенное равноправное местоположение для очистки. Описанный механизм является специфическим для реализации, но должен выполняться аппаратными средствами без участия ПО [7].

Блокированные транзакции

Поддержка задержанных транзакций необходима для предотвращения блокировки в системе при использовании унаследованного ПО, которое инициирует доступы к устройствам ввода-вывода. Некоторые процессоры могут генерировать блокированные доступы как результат выполнения инструкций, которые неявно блокируют счетчик. Некоторое старое ПО неправильно применяет эти транзакции и генерирует блокированные последовательности, даже когда монопольный доступ не требуется. Поскольку блокированные доступы к устройствам ввода-вывода приводят к потенциальным блокировкам помимо упомянутых выше, что может привести к серьезному снижению производительности, то для оконечных устройств PCI Express запрещена поддержка монопольных доступов. Новое ПО также не должно использовать инструкции, которые будут инициировать монопольные доступы к устройствам ввода-вывода. Поддержка оконечными устройствами монопольных доступов введена только из-за вопросов совместимости с существующим ПО.

Инициирование блокированных запросов PCI Express разрешено только корневому комплексу. Блокированные запросы, инициированные оконечными уст-

ройствами и мостами, не поддерживаются. Данная непротиворечивость с ограничениями для блокированных транзакций использует принципы спецификации *"PCI Local Bus Specification" версии 2.3*.

Данный раздел определяет правила, связанные с поддержкой монопольных доступов от главного процессора к устройствам Legacy Endpoint, включая распространение таких транзакций через коммутаторы и мосты PCI Express/PCI [7].

Правила инициирования и распространения блокированных транзакций

Блокированные последовательности генерируются главным процессором(ми) как одна или больше операций чтения, следующих за равным количеством записей в ту же область(ти). Когда установлен монопольный доступ, весь остальной трафик блокируется от использования пути между корневым комплексом и заблокированным устройством Legacy Endpoint или мостом.

- ❒ Блокировка инициируется на PCI Express, используя тип "lock" — Read Request/Completion (MRdLk/CplDLk) и завершается с сообщением "Unlock".
  - Семантика MRdLk, CplDLk и Unlock разрешена только для класса трафика, определенного по умолчанию (TC0).
- ❒ Сообщение "Unlock" является широковещательным из корневого комплекса для всех оконечных устройств и мостов.
  - Любое устройство, которое вовлечено в блокированную последовательность, должно игнорировать это сообщение.

Введение и распространение блокированной транзакции через PCI Express выполняется следующим образом:

- ❒ Последовательности блокированных транзакций начинаются с запроса MRdLk.
  - Любые последовательные чтения для блокированных транзакций также используют запросы MRdLk.
  - Выполнения для любого запроса MRdLk используют тип выполнения CplDLk.

Оконечные устройства Legacy Endpoint

Устройствам Legacy Endpoint запрещено поддерживать монопольные доступы, хотя и допускается их использование. Если монопольный доступ поддержан, устройство должно обрабатывать его в соответствии со следующими правилами.

- ❒ Устройство становится блокированным, когда оно передает первое выполнение для первого запроса чтения монопольного доступа.

- Один раз заблокированное устройство должно оставаться в этом состоянии, пока оно не примет сообщения "Unlock".
- При нахождении в заблокированном состоянии устройство не должно генерировать никакого запроса, использующего классы трафика, которые отображаются в виртуальный канал по умолчанию (VC0).

  Данное требование применяется ко всем возможным источникам запросов внутри оконечного устройства, в случае, когда внутри него присутствуют больше чем один возможный источник.
- Запросы могут быть сгенерированы путем использования класса трафика, отображаемого в виртуальный канал, отличный от виртуального канала по умолчанию.

Оконечные устройства PCI Express

Оконечные устройства PCI Express не поддерживают монопольный доступ. Данные устройства должны интерпретировать запросы MRdLk как неподдерживаемые запросы.

Правила сброса для PCI Express

Данный раздел определяет поведение канала PCI Express при сбросе. Сброс может быть сгенерирован платформой или на компоненте, но любые отношения между сбросом канала PCI Express и сбросом компонента или платформы являются специфическими для компонента или платформы соответственно.

- Должен присутствовать аппаратный механизм для установления или возвращения всех состояний порта в начальные условия, определенные стандартом — этот механизм называется *Power Good Reset*.
  - Сброс Power Good Reset, происходящий после подачи питания к компоненту, называется *холодным сбросом* или, иначе, *Cold Reset*.
  - В некоторых случаях возможен запуск механизма "Power Good Reset" аппаратным обеспечением без снятия и подачи питания компонента. Такой сброс называется *теплым сбросом* или, иначе, *Warm Reset*.
  - Также существует внеполосный механизм для распространения сброса за пределами канала, он называется *горячим сбросом* или, иначе, *Hot Reset*.
  - Переход в состояние "DL_Inactive" в некоторых случаях идентичен сбросу Hot Reset.
- При выходе из любого типа сброса (Cold, Warm или Hot), все регистры порта и конечные автоматы должны быть установлены в их начальные состояния, определенные стандартом PCI Express.

- При выходе из состояния Power Good Reset физический уровень будет пытаться запустить ("поднять") канал. Как только оба компонента вошли в состояние начальной проверки канала, то далее их состояние будет изменяться через инициализацию канала для физического уровня и затем через инициализацию для виртуального канала VC0, подготавливая таким образом уровень транзакций и канальный уровень к использованию канала.

  После инициализации VC0 пакеты TLP и DLLP могут быть переданы через канал.

После сброса некоторые устройства потребуют дополнительного времени, перед тем как они будут способны ответить на принятые запросы. Для конфигурационных запросов особенно необходимо, чтобы компоненты и устройства вели себя детерминистическим путем, который следует правилам адресации.

Правила требований адресации для компонентов и устройств разделяются на два подмножества: требования к компонентам и требования к системе.

Требования правил адресации к компонентам.

- Компонент должен входить в начальное активное состояние проверки канала в пределах 80 мс после завершения сброса Power Good Reset.

  В некоторых системах возможно, что два компонента на канале могут выйти из состояния Power Good Reset в разное время. Каждый компонент должен соблюдать требования для входа в начальное активное состояние проверки канала в пределах 80 мс после окончания Power Good Reset с их точки зрения, т. е. после того, как они определили, что произошел выход из сброса.

- При выполнении проверки канала (вхождении в состояние "DL_Active") компонент должен быть способен принять и обработать пакеты TLP и DLLP.

Требования правил адресации к системе.

- В целях корректного выполнения компонентом внутренней инициализации системное ПО должно ожидать минимум 100 мс после окончания сброса (Cold/Warm/Hot), перед тем как разрешать генерирование конфигурационных запросов к устройствам PCI Express.

  Система должна гарантировать, что все компоненты, предназначенные для взаимодействия с ПО во время загрузки, готовы принять конфигурационные запросы в пределах 100 мс после окончания состояния Power Good Reset; реализация не определена стандартом и возлагается на разработчика.

- Корневой комплекс и/или системное ПО после сброса (Hot/Warm/Cold) должны ожидать от устройства возвращения статуса "Successful Comp-

letion" для правильного конфигурационного запроса 1.0 с. По истечении этого времени устройство считается неисправным.

- Если корневой комплекс повторяет конфигурационные запросы, завершенные со статусом "Configuration Request Retry", тогда они должны быть повторены до 1 с после истечения времени T_{0RC} (время задержки корневого комплекса), в этой точке корневой комплекс может завершить запрос как неподдерживаемый.
- Данная задержка аналогична параметру T_{rhfa}, определенному для PCI/PCI-X, и предназначена для выделения адекватного количества времени устройству, которому необходимо провести собственную инициализацию.

❒ При попытке конфигурационного доступа к устройствам в PCI или PCI-X сегменте позади PCI Express/PCI(-X) моста временной параметр T_{rhfa} должен соответствовать данным стандартам.

Правила при нормальном функционировании канала.

❒ Если для каких-то целей понижается нормальное функционирование канала, то канальный уровень и уровень транзакций войдут в состояние "DL_Inactive".

❒ Для любого виртуального или реального PCI-моста любое из следующих событий вызывает сброс вторичной стороны моста, используя механизм физического уровня для сообщения сброса канала:

- установка бита **Secondary Bus Reset** в регистре Bridge Control;
- переход в состояние "DL_Inactive" на основной стороне моста;
- сброс канала с использованием механизма физического уровня для сообщения "Link Reset".

Определенные аспекты сброса Power Good Reset, описанные в стандарте PCI Express, являются специфическими для реализации платформы и форм-фактора. Определенные платформы, форм-факторы или приложения могут требовать дополнительной спецификации временных и/или отношений порядка между компонентами системы для Power Good Reset. Например, они могут требовать, чтобы все компоненты PCI Express внутри шасси соблюдали переход в состояние Power Good Reset и выход из него в одно и то же время (в пределах некоторых допусков). В многошассийном окружении для выхода из состояния Power Good Reset может потребоваться, чтобы корпус, содержащий корневой комплекс, был последним [7].

Во всех случаях, когда подано питание, должны быть определены следующие временные параметры:

❒ T_{pvpgl} — минимальное время, в течение которого сигнал Power_Good должен оставаться неактивным после подачи питания;

- T_{pwrgd} — минимальное время, в течение которого сигнал Power_Good должен находиться в неактивном состоянии после снятия;
- T_{fail} — после подачи питания сигнал Power_Good должен быть снят внутри этого временного интервала.

Кроме того, могут быть определены и дополнительные параметры.

Во всех случаях, когда поддержан генератор синхроимпульсов, должен быть определен следующий параметр:

- $T_{pwrgd\text{-}clk}$ — минимальное время, в течение которого сигнал Power_Good должен оставаться неактивным, после того как любой поддерживаемый генератор перейдет в стабильное состояние.

Также при необходимости могут быть определены дополнительные параметры [7].

Поддержка механизма Hot-Plug

Архитектура PCI Express разработана с собственной поддержкой горячего подключения и горячей замены устройств (Hot-Plug). В данном подразделе определена стандартная пользовательская модель для всех форм факторов PCI Express, поддерживающих "горячую" замену и "горячее" подключение устройств. Эта модель предоставляет основу для поведения индикаторов и кнопок, если они реализованы в системе. Определение индикаторов и кнопок применяется ко всем моделям PCI Express Hot-Plug [7].

Пользовательская модель PCI Express Hot-Plug

Стандартная пользовательская модель, как следует из названия, в первую очередь нацелена на пользователей, которые эксплуатируют системы со слотами Hot-Plug. Такое назначение объясняется тем, что обычно в пользовательских системах присутствует аппаратное и программное обеспечение от различных производителей. Модель позволяет пользователям использовать слоты Hot-Plug их систем без дополнительного переобучения. Стандартная пользовательская модель PCI Express Hot-Plug выведена из стандартной пользовательской модели, определенной в спецификации *"PCI Standard Hot-Plug Controller and Subsystem Specification" версии 1.0.* Данные модели идентичны с точки зрения пользователя. Были произведены лишь незначительные изменения в определениях регистров и соответствия со стандартной пользовательской моделью, требуемые всеми форм-факторами PCI Express, которые реализуют Hot-Plug и используют индикаторы и кнопки [7].

Девиация форм-факторов PCI

Отклонение от стандартной пользовательской модели приводит к несовместимым с PCI-Express решениям и будет проявляться в нежелательных результатах, таких как:

- ❒ сложность эксплуатации для пользователя;
- ❒ более дорогое тестирование аппаратного обеспечения;
- ❒ функциональная несовместимость с системным ПО;
- ❒ ошибки системного ПО из-за непроверенного поведения системы.

Элементы стандартной пользовательской модели

В табл. 12.8 приведены элементы модели с описанием, а подробное описание представлено далее.

Таблица 12.8. *Элементы стандартной пользовательской модели*

Элемент	Цель, назначение
Индикаторы	Показывают состояние слота: состояние питания и готовность к операции извлечения
Ручная защелка MRL (Manually-operated Retention Latch)	Удерживает плату расширения на месте
Сенсор защелки MRL	Позволяет порту и системному ПО определить открытие защелки MRL
Электромеханическая блокировка	Предотвращает вынимание платы расширения, если в слоте присутствует напряжение
Кнопка внимания — Attention Button	Позволяет пользователю запрашивать операции Hot-Plug
Программный пользовательский интерфейс — Software User Interface	Позволяет пользователю запрашивать операции Hot-Plug
Нумерация слотов	Обеспечивает визуальную идентификацию слотов

Индикаторы

Стандартная пользовательская модель определяет два индикатора: индикатор питания и индикатор внимания. Платформа может обеспечить два индикатора в каждый слот или панель модуля, индикаторы могут быть реализованы на корпусе или модуле, детали реализации зависят от требований форм-фактора "горячего" подключения. Каждый индикатор находится в одном из трех состояний: вкл., выкл. или мерцание. Системное ПО Hot-Plug обладает исклю-

чительным контролем над состоянием индикаторов за счет возможности записи в командный регистр, связанный с индикатором.

Порт совместимый с Hot-Plug управляет частотой мерцания индикаторов, рабочим циклом и фазой. Мерцающие индикаторы функционируют на частоте от 1 до 2 Гц с коэффициентом заполнения 50% (± 5%). Мерцающие индикаторы не должны быть синхронизированы и синфазны между портами.

Индикаторы должны находиться в непосредственной близости от связанного с ними слота Hot-Plug, если индикаторы реализованы на корпусе, чтобы соединение между индикаторами и слотом Hot-Plug было как можно более свободным.

Оба индикатора полностью контролируются системным ПО. Устройство коммутатора или корневого порта никогда не изменяет состояние индикатора при отклике на событие, типа сбоя питания или внезапного открытия защелки MRL, если только системное ПО специально не пошлет такую команду. Исключение предоставляется платформам, которые совместимы с механизмом определения контактной неисправности (типа "залипания") питания. В этом специфическом случае сбоя платформе разрешено "подавить" устройство коммутатора или корневого порта и силой включить индикатор питания (как указание, что плата расширения не может быть извлечена). Во всех случаях внутреннее состояние порта для индикатора питания должно соответствовать состоянию, выбранному программным обеспечением. Обработка системным ПО константных неисправностей является необязательной функциональностью и отдельно не описывается. Поэтому производитель платформы должен гарантировать, что эта дополнительная функциональность стандартной пользовательской модели выполняется дополнительным ПО, описывается в документации платформы или каким-либо другим способом.

Индикатор внимания

Индикатор внимания "Attention" желтого или янтарного цвета используется для указания на проблемы функционирования или указывает, что слот Hot-Plug находится в процессе идентификации, что позволяет локализовать его состояние.

Таблица 12.9. *Состояния индикатора внимания "Attention"*

Состояние индикатора	Значение
Выкл.	Нормальное функционирование
Вкл.	В этом слоте есть проблемы функционирования
Мерцание	Слот идентифицируется на запрос пользователя

Индикатор внимания выключен

Когда индикатор внимания "Attention" выключен, это означает, что плата расширения (если таковая представлена) и слот Hot-Plug функционируют нормально и не требуют внимания.

Индикатор внимания включен

Если индикатор внимания "Attention" включен, это означает, что существуют проблемы функционирования платы или слота.

Под проблемами функционирования понимаются условия, которые предотвращают продолжение функционирования платы расширения. Операционная система или другое системное ПО определяет это состояние платы расширения через состояние соответствующего индикатора внимания. Примерами проблем функционирования могут служить неполадки внешнего кабеля, платы расширения, программных драйверов и сбои питания. В общем случае включенное состояние индикатора внимания означает, что была предпринята попытка функционирования, и она была неудачной, или что произошло непредвиденное событие.

Индикатор внимания не используется для сообщения о проблемах, обнаруженных при проверке достоверности запроса для операции Hot-Plug. Термин "проверка достоверности" применяется к любой проверке, которую выполняет системное ПО для гарантии, что запрошенная операция не вызовет проблем. Примерами сбоя проверки достоверности может быть отказ в выполнении операции Hot-Plug, неудовлетворительное распределение питания, и другие состояния, которые могут быть обнаружены перед началом операции.

Мерцание индикатора внимания

Мерцание индикатора внимания "Attention" означает, что системное ПО идентифицирует данный слот при запросе оператора. Данное поведение управляется пользователем (например, через программный интерфейс пользователя или средства управления).

Индикатор питания

Индикатор питания имеет приятный зеленый цвет и используется для указания состояния питания слота (табл. 12.10).

Таблица 12.10. *Состояния индикатора питания*

Состояние индикатора	Значение
Выкл.	Питание снято. Разрешена установка или извлечение платы расширения. Все напряжения источников питания (кроме Vaux) были сняты из слота при необходимости извлечения платы. Напряжение Vaux снимается, когда открыта защелка MRL

Таблица 12.10 *(окончание)*

Состояние индикатора	Значение
Вкл.	Питание подано. Установка или извлечение платы расширения не разрешено
Мерцание	Переход питания. Слот находится в процессе подачи или снятия питания. Установка или вынимание платы расширения не разрешено

Индикатор питания отключен

Когда индикатор питания выключен, это означает, что разрешена установка или извлечение платы расширения. Основное питание слота снято, если это необходимо для форм-фактора, примером снятия основного питания является форм-фактор платы PCI Express. Если платформа обеспечивает напряжение Vaux в слоты Hot-Plug и защелка MRL закрыта, то любые сигналы, коммутируемые защелкой MRL, сообщаются в слот независимо от состояния индикатора. При открытии защелки MRL коммутируемые ей сигналы снимаются. Системное ПО должно отключать индикатор питания, когда слот не запитан и/или разрешено вставлять или извлекать платы расширения. Данное правило диктуется соответствующей электромеханической спецификацией на форм-фактор.

Индикатор питания включен

Когда индикатор питания включен, это означает, что основное питание подано в слот и что установка или извлечение платы расширения запрещено.

Мерцание индикатора питания

Мерцание индикатора питания означает, что в слот подается питание или снимается из него и что установка или извлечение плат расширения запрещено. Мерцание индикатора питания также обеспечивает оператора зрительной обратной связью при нажатии кнопки "Attention Button".

Ручная защелка MRL

Защелка MRL (Manually-operated Retention Latch) — это управляемый вручную механизм удержания, который удерживает плату расширения в слоте и предотвращает извлечение платы пользователем. Эта защелка жестко держит плату в слоте, так что кабели могут быть присоединены без риска создания прерывистого контакта. В платформах, которые не реализуют сенсоры MRL, разрешены защелки MRL, которые держат две или больше плат расширения одновременно [7].

Сенсор MRL

Сенсором MRL может быть коммутируемое, оптическое или другое сенсорное устройство, которое сообщает порту позицию защелки MRL. Сенсор MRL сообщает "закрыто", когда защелка MRL полностью закрыта, и "открыто" во всех других случаях (т. е. полностью открыта и все промежуточные положения).

Если питание Vaux подведено к слотам Hot-Plug, то коммутируемые защелкой MRL сигналы должны быть автоматически сняты со слота, если сенсор MRL указывает, что защелка MRL открыта, и сигналы должны быть поданы в слот, если сенсор MRL указывает, что защелка MRL была снова закрыта.

Сенсор MRL позволяет порту отслеживать позицию защелки MRL и, следовательно, позволяет определить внезапное открытие защелки MRL. Когда внезапное открытие защелки MRL связано с уже идентифицированным слотом, то порт изменяет состояние этого слота на выключенное и уведомляет системное ПО. Порт не изменяет состояние индикаторов питания и внимания [7].

Электромеханическая блокировка

Электромеханическая блокировка — это механизм для физического блокирования платы расширения или защелки MRL до тех пор, пока системное ПО и порт не освободят ее. Реализация блокировки необязательна, а в программном интерфейсе отсутствует механизм для точного управления электромеханической блокировкой. Стандартная пользовательская модель предполагает, что если электромеханические блокировки реализованы, то они управляют одним выходным сигналом порта, который подает основное питание в слот. Системы могут дополнительно расширять управление блокировками для обеспечения физической защиты плат расширения [7].

Кнопка внимания

Кнопка внимания (Attention Button) является кнопкой мгновенного срабатывания, размещенной рядом с каждым слотом Hot-Plug или на модуле, и управляется пользователем для начала операции "горячего" извлечения или замены в данном слоте.

Индикатор питания обеспечивает визуальную обратную связь с оператором (если системное ПО принимает запрос, инициированный кнопкой внимания) путем мерцания. Как только индикатор питания замерцал, то пользователю дается пятисекундный интервал времени на отмену, во время которого повторное нажатие кнопки отменяет операцию.

Если операция, инициированная кнопкой, срывается по любым причинам, то рекомендуется, чтобы системное ПО представило сообщение, объясняющее

сбой, через программный пользовательский интерфейс или добавило его в системный журнал [7].

Программный интерфейс пользователя

Системное ПО должно обеспечивать *программный пользовательский интерфейс* — Software User Interface, который позволяет запустить операции горячего извлечения и замены и который позволяет отследить состояние захваченного порта. Детальное рассмотрение пользовательского интерфейса горячего подключения является спецификой ОС и поэтому не определено в стандарте PCI Express.

В системах с несколькими слотами Hot-Plug системное ПО должно позволять пользователю инициировать операции в каждом слоте независимо от состояния остальных слотов. Таким образом, пользователю разрешено инициировать операции Hot-Plug в слоте, используя либо программный пользовательский интерфейс, либо кнопку внимания (Attention Button), пока на другом слоте происходит операция Hot-Plug, независимо от того, какой интерфейс был использован для начала первой операции [7].

Функция распределения питания

С добавлением возможности горячей замены для плат расширения возникает необходимость в способности системы правильно выделить питание для любого нового устройства, добавленного в систему. Данная функциональная возможность отделена от управления питанием; базовый уровень поддержки должен гарантировать правильное функционирование системы. Концепция распределения питания допускает блочное (узловое) построение узлов, что позволяет устройствам взаимодействовать с системой для достижения перечисленных ранее целей. Существует множество путей, которыми возможно реализовать в системе функциональности управления питанием, и как они выходят за пределы стандарта.

Устройства, которые будут представлены на платах расширения Hot-Plug, должны реализовывать функции распределения питания. Устройства, предназначенные для использования на платах расширения или материнских платах, имеют опцию поддержки функции распределения питания. Устройства, которые разработаны и для плат расширения, и для модулей, должны также реализовывать распределение питания. Стандарт PCI Express требует, чтобы устройства и/или платы расширения превышали допустимый предел "конфигурационного" питания, определенный в соответствующей электромеханической спецификации, до окончания процесса конфигурирования и их включения системой. Системы должны правильно распределять питание до включения плат расширения [7].

Рекомендации процесса системного распределения питания

Рекомендуется, чтобы системное встроенное ПО предоставляло агенту управления распределением питания следующую информацию:

- ❒ полное питание системы (информация об источнике питания);
- ❒ полное питание, выделенное системным встроенным ПО (устройствами материнской платы);
- ❒ полное количество слотов и типов слотов.

Системное встроенное ПО отвечает за выделение питания для всех устройств на материнской плате, которые не имеют функций распределения питания. Встроенное ПО может как охватывать стандартные устройства PCI Express, которые соединены со стандартными шинами питания, так и не выделять питание для них. При выделении встроенным ПО питания для устройства, ПО должно установить бит **SYSTEM_ALLOC** регистра Power Budget Capability (регистр устройства) в состояние логической "1", для указания успешности операции. Менеджер распределения питания отвечает за выделение питания всем устройствам PCI Express, в том числе устройствам материнской платы, имеющим функции распределения питания, но которые были помечены для выделения. Менеджер распределения питания также отвечает за определение возможности подключения устройств Hot-Plug в системе.

Указанные методы могут обеспечить одинаковую функциональность, и не требуется, чтобы процесс распределения питания был реализован именно в этой манере [7].

Управление ограничением питания слота

PCI Express предоставляет механизм для программно-управляемого ограничения максимальной мощности в каждый слот, которую может потреблять плата/модуль PCI Express (связанные с этим слотом). Ключевыми элементами этого механизма являются:

- ❒ поля **Slot Power Limit Value** и **Scale** регистра Slot Capability, реализованного в Downstream-портах корневого комплекса и коммутатора;
- ❒ поля **Slot Power Limit Value** и **Scale** регистра Device Capability, реализованного в Upstream-портах оконечного устройства, коммутатора и моста PCI Express-to-PCI;
- ❒ сообщение "Set_Slot_Power_Limit". Это сообщение передает содержимое полей **Slot Power Limit Value** и **Scale** регистра Slot Capability порта Downstream (корневого комплекса или коммутатора) в соответствующие поля **Slot Power Limit Value** и **Scale** регистра Device Capability порта Upstream компонента, присоединенного к тому же каналу.

Пределы потребляемой мощности на платформе обычно контролируются ПО (например, встроенным ПО платформы), которое учитывает специфику платформы, такую как:

- ❒ разделение платформы, включая слоты для расширения ввода-вывода, использующие платы/модули расширения;
- ❒ возможности по обеспечению питанием;
- ❒ температурные возможности.

Данное ПО отвечает за корректное программирование полей **Slot Power Limit Value** и **Scale** регистров Slot Capability портов Downstream, соединенных со слотами расширения. После того как значение было записано в регистр внутри Downstream-порта, оно передается к другому компоненту, соединенному с этим портом путем сообщения "Set_Slot_Power_Limit". Получатель должен использовать содержащееся в сообщении значение для ограничения использования питания всей платы/модуля. Исключение составляют платы/модули, которые ни при каких условиях не выходят за предел минимального значения, определенного в соответствующей электромеханической спецификации. Предполагается, что программное обеспечение драйвера устройства платы/модуля будет в состоянии (путем чтения значения полей **Slot Power Limit Value** и **Scale** регистра Device Capability) отконфигурировать аппаратное обеспечение платы/модуля таким образом, что плата/модуль не превысят продиктованный предел. В случае, когда платформа определяет предел, который ниже минимума, необходимого для нормального функционирования, драйвер устройства должен быть в состоянии сообщить данное несоответствие верхнему уровню конфигурационного ПО.

Следующие правила относятся к механизму управления "Slot Power Limit":

Правила для плат/модулей

- ❒ До тех пор, пока не будет принято сообщение "Set_Slot_Power_Limit", указывающее значение предела, большее чем минимальное значение, определенное в электромеханической спецификации для форм-фактора платы или модуля, плата/модуль не должны потреблять питания больше, чем определено этим минимальным значением.
- ❒ Максимально допустимая потребляемая мощность для платы/модуля определяется самым большим значением из всех принятых сообщений "Set_Slot_Power_Limit".
- ❒ Оконечным устройствам, коммутатору и мосту PCI Express-to-PCI, которые предназначены для объединения на плате/модуле, где полное потребление мощности ниже минимального предела для данного форм-фактора, запрещено игнорировать сообщения "Set_Slot_Power_Limit" и возвращать значение "0" в полях **Slot Power Limit Value** и **Scale** регистра Device Capability.

- Перечисленные ранее компоненты должны корректно принять сообщение "Set_Slot_Power_Limit", но вместо обработки просто отменить его.

Правило для корневых комплексов и коммутаторов, содержащих слоты

- Downstream-порт не должен передавать сообщение "Set_Slot_Power_Limit", предел которого будет меньше, чем минимальное значение, определенное в электромеханической спецификации для форм-фактора этих слотов.

Управляющие регистры Slot Power Limit

Обычно регистры Slot Power Limit внутри Downstream-портов корневого комплекса или коммутатора программируются специфическим программным обеспечением платформы. Некоторые реализации могут использовать аппаратный метод для инициализации значения этих регистров и таким образом не требуют программной поддержки.

Оконечные устройства, коммутатор и мост "PCI Express-to-PCI", предназначенные для объединения на плате/модуле, где полная потребляемая мощность ниже минимального предела, определенного для данного форм-фактора, могут игнорировать сообщения "Set_Slot_Power_Limit". Компоненты PCI Express, реализованные подобным образом, могут быть не совместимы с потенциальными будущими форм-факторами. Такие форм-факторы возможно будут сообщать более низкий предел потребляемой мощности, чем минимально необходимый для новой платы/модуля, разработанной на существующих компонентах [7].

Заключение

Шина PCI прослужила более десяти лет и за это время зарекомендовала себя наилучшим образом. Такие качества, как надежность, эффективность и гибкость в сочетании с достаточной пропускной способностью, обеспечили ее использование в качестве локальной шины с последовательной архитектурой. При этом так и не были широко внедрены возможности шины по расширению и наращиванию. Современные требования к пропускной способности и возможностям шины обусловили вытеснение последовательной локальной шины, на ее место встала параллельная шина с хабовой архитектурой — PCI Express. Тем не менее переход будет происходить достаточно долго, в первую очередь из-за огромного количества уже разработанных устройств на шине PCI. По этим же причинам долгое время будет обеспечиваться обратная совместимость шины PCI Express с шиной PCI.

Однако не следует сбрасывать со счетов стандарт, который работает уже более десяти лет, идеи и принципы, заложенные в него, могут быть использованы в решениях, не требовательных к пропускной способности. Шина PCI не отходит в прошлое, она изменяет свое назначение и, возможно, воскреснет в других устройствах.

С автором книги можно связаться по адресу издательства "БХВ-Петербург" **mail@bhv.ru**.

ПРИЛОЖЕНИЕ 1

Идентификаторы функциональностей

В данном приложении перечислены существующие идентификаторы функциональных возможностей — "Capability ID" (табл. П1.1). Каждая функциональность должна иметь назначенный PCI SIG код идентификации ID. Эти коды назначаются и обрабатываются так же, как и коды классов. В *главе 11* приведено полное описание расширенного механизма функциональностей для PCI-устройств [9].

Таблица П1.1. *Идентификаторы функциональностей*

ID	Функциональность
0	Зарезервировано
1	PCI Power Management Interface (PMI). Эта функциональность обеспечивает стандартный интерфейс для управления функциями управления питанием в PCI-устройстве
2	AGP. Эта функциональность идентифицирует контроллер, который способен использовать функции порта AGP (Accelerated Graphics Port)
3	VPD. Эта функциональность идентифицирует устройство, которое поддерживает VPD (Vital Product Data)
4	Slot Identification. Эта функциональность идентифицирует мост, который обеспечивает внешние расширенные возможности
5	Message Signaled Interrupts (MSI). Эта функциональность идентифицирует функцию PCI, которая может доставлять сообщение о прерывании
6	CompactPCI Hot Swap. Эта функциональность обеспечивает стандартный интерфейс для управления и опознания статуса внутри устройства, которое поддерживает "горячую" замену в системе CompactPCI
7	PCI-X

Таблица П1.1 *(окончание)*

ID	Функциональность
8	Зарезервировано для AMD
9	Vendor Specific. Этот ID-код позволяет производителям устройств использовать механизм функциональностей для специфической информации производителя. Размещение информации возлагается на производителя, за исключением байта, немедленно следующим за указателем "Next" в структуре функциональности, который определен как поле длины (или размера). Это поле длины обеспечивает количество байтов в структуре (включая байты ID-кода и указателя Next). Примером специфического использования производителем является устройство, которое конфигурируется на последних заводских этапах как 32- или 64-битный агент, и определенная производителем структура сообщает драйверу устройства о том, какие функции поддерживает устройство
0xA	Отладочный порт
0xB	Управление центральным ресурсом CompactPCI
0xC	PCI Hot-Plug. Этот ID-код указывает, что связанное с ним устройство совместимо с моделью стандарта контроллера Hot-Plug
0xD—0xFF	Зарезервировано

ПРИЛОЖЕНИЕ 2

Коды классов

В данном приложении приведены текущие значения кодов класса. Список может быть расширен в любое время, последние версии содержатся на сайте PCI SIG (**www.pcisig.com**). Компании, желающие определить новые значения, должны связаться с PCI SIG. Все неопределенные значения являются зарезервированными для назначения SIG [9]. В табл. П2.1 представлены коды базовых классов, далее каждый класс описан отдельно.

Таблица П2.1. *Базовые классы*

Базовый класс	Значение
00h	Устройство было разработано до определения кодов классов
01h	Контроллер накопителей данных
02h	Сетевой контроллер
03h	Графический контроллер
04h	Устройство мультимедиа
05h	Контроллер памяти
06h	Устройство мост
07h	Простой коммуникационный контроллер
08h	Основная системная периферия
09h	Устройство ввода
0Ah	Базовый блок
0Bh	Процессор
0Ch	Контроллер последовательной шины

Таблица П2.1 *(окончание)*

Базовый класс	Значение
0Dh	Беспроводной контроллер
0Eh	Интеллектуальный контроллер ввода-вывода
0Fh	Контроллер спутниковой связи
10h	Контроллер шифрации/дешифрации
11h	Контроллер приема и обработки данных
12h—FEh	Зарезервировано
FFh	Устройство не согласовывается ни с одним из определенных классов

Базовый класс 00h

Этот базовый класс определен для обратной совместимости с устройствами, которые были разработаны до определения поля кода класса **Class Code**. Новые устройства не будут использовать это значение и существующие устройства должны по возможности переключаться на более подходящее значение. Для кодов класса с этим значением базового класса, определено два значения, как показано в табл. П2.2.

Таблица П2.2. *Значения базового класса 0*

Базовый класс	Подкласс	Интерфейс	Значение
00h	00h	00h	Все существующие реализованные устройства, кроме VGA-совместимых
	01h	00h	VGA-совместимое устройство

Базовый класс 01h

Этот базовый класс определен для всех типов контроллеров накопителей данных. Табл. П2.3 содержит все возможные значения для данного класса. Имеется несколько значений подклассов. Только для подкласса "Контроллер IDE" определен специфический программный интерфейс на уровне регистров, размещение байтов которого приведено на рис. П2.1.

Рис. П2.1. Размещение байтов программного интерфейса для кода класса контроллера IDE

Таблица П2.3. *Значения базового класса 1*

Базовый класс	Подкласс	Интерфейс	Значение
01h	00h	00h	Контроллер шины SCSI
	01h	xxh	Контроллер IDE
	02h	00h	Контроллер FDD
	03h	00h	Контроллер шины IPI
	04h	00h	Контроллер RAID
	05h	20h	Контроллер ATA с одним каналом DMA
		30h	Контроллер ATA со сцепленным DMA
	80h	00h	Контроллер накопителя данных другого типа

Базовый класс 02h

Данный базовый класс определен для всех типов сетевых контроллеров. Определены несколько подклассов. Программный интерфейс на уровне регистров не назначен. Табл. П2.4 содержит все возможные значения для данного класса.

Таблица П2.4. *Значения базового класса 2*

Базовый класс	Подкласс	Интерфейс	Значение
02h	00h	00h	Контроллер Ethernet
	01h	00h	Контроллер Token Ring
	02h	00h	Контроллер FDDI
	03h	00h	Контроллер ATM
	04h	00h	Контроллер ISDN
	05h	00h	Контроллер WorldFip
	06h	xxh*	PICMG 2.14 Multi Computing
	80h	00h	Контроллер других сетей

* Для получения информации об использовании этого поля см. PICMG 2.14 Multi Computing Specification (**http://www.picmg.com**).

Базовый класс 03h

Данный базовый класс определен для всех типов графических контроллеров. Для VGA-устройств (подкласс 00h) байт программируемого интерфейса разделен на битовые поля, которые идентифицируют дополнительные возможности видеоконтроллера. Устройство может поддерживать несколько интерфейсов путем использования битовой карты для указания поддерживаемых интерфейсов. Для устройства XGA (подкласс 01h) определен только интерфейс стандарта XGA. Подкласс 02h определен для контроллеров, которые имеют аппаратную поддержку для 3D-операций и не совместимы с VGA. Табл. П2.5 содержит все возможные значения для данного класса.

Таблица П2.5. *Значения базового класса 3*

Базовый класс	Подкласс	Интерфейс	Значение
03h	00h	00000000b	VGA-совместимый контроллер. Память адресуется в диапазоне от 0A0000h до 0BFFFFh. Пространство ввода-вывода адресуется от 3B0h до 3BBh и от 3C0h до 3DFh, включая все псевдонимы этих адресов
		00000001b	8514-совместимый контроллер: адресуемый 2E8h и его псевдонимы, 2EAh—2EFh

Таблица П2.5 *(окончание)*

Базовый класс	Подкласс	Интерфейс	Значение
	01h	00h	Контроллер XGA
	02h	00h	Контроллер 3D
	80h	00h	Графический контроллер другого типа

Базовый класс 04h

Данный базовый класс определен для всех типов мультимедийных устройств. Имеется несколько значений подклассов. Программный интерфейс на уровне регистров не определен. Табл. П2.6 содержит все возможные значения для данного класса.

Таблица П2.6. *Значения базового класса 4*

Базовый класс	Подкласс	Интерфейс	Значение
04h	00h	00h	Графическое устройство
	01h	00h	Звуковое устройство
	02h	00h	Устройство компьютерной телефонии
	80h	00h	Мультимедийное устройство другого типа

Базовый класс 05h

Данный базовый класс определен для всех типов контроллеров памяти. Имеется несколько значений подклассов. Программный интерфейс на уровне регистров не определен. Табл. П2.7 содержит все возможные значения для данного класса.

Таблица П2.7. *Значения базового класса 5*

Базовый класс	Подкласс	Интерфейс	Значение
05h	00h	00h	Контроллер RAM
	01h	00h	Контроллер Flash
	80h	00h	Контроллер памяти другого типа

Базовый класс 06h

Данный базовый класс определен для всех типов мостовых устройств. PCI-мостом является любое PCI-устройство, которое отображает PCI-ресурсы (память или I/O) с одной стороны устройства к другой. Имеется несколько значений подклассов. Табл. П2.8 содержит все возможные значения для данного класса.

Таблица П2.8. *Значения базового класса 6*

Базовый класс	Подкласс	Интерфейс	Значение
06h	00h	00h	Главный мост
	01h	00h	Мост ISA
	02h	00h	Мост EISA
	03h	00h	Мост MCA
	04h	00h	Мост PCI-to-PCI
		01h	Мост PCI-to-PCI с вычитающей дешифрацией. Данный код интерфейса идентифицирует мост PCI-to-PCI как устройство, которое поддерживает вычитающую дешифрацию в дополнение ко всем определенным в настоящее время функциям моста PCI-to-PCI
	05h	00h	Мост PCMCIA
	06h	00h	Мост NuBus
	07h	00h	Мост CardBus
	08h	xxh	Мост RACEway (см. далее)
	09h	40h	"Полупрозрачный" мост PCI-to-PCI с основной стороны шины PCI, обращенной к главному процессору
		80h	"Полупрозрачный" мост PCI-to-PCI со вторичной стороны шины PCI, обращенной к главному процессору
	0Ah	00h	Основной мост InfiniBand-to-PCI
	80h	00h	Мост другого типа

RACEway является стандартом ANSI (ANSI/VITA 5-1994) структуры коммутации. Биты [07:01] программного интерфейса зарезервированы, доступны

только для чтения и возвращают ноль. Бит 0 доступен только для чтения и определяет режим функционирования:

- ❒ прозрачный режим;
- ❒ режим конечной точки.

Базовый класс 07h

Данный базовый класс определен для всех типов простых коммуникационных контроллеров. Имеется несколько значений подклассов, некоторые из которых располагают специфическим известным программным интерфейсом на уровне регистров. Табл. П2.9 содержит все возможные значения для данного класса.

Таблица П2.9. *Значения базового класса 7*

Базовый класс	Подкласс	Интерфейс	Значение
07h	00h	00h	XT-совместимый последовательный контроллер общего типа
		01h	16450-совместимый последовательный контроллер
		02h	16550-совместимый последовательный контроллер
		03h	16650-совместимый последовательный контроллер
		04h	16750-совместимый последовательный контроллер
		05h	16850-совместимый последовательный контроллер
		06h	16950-совместимый последовательный контроллер
	01h	00h	Параллельный порт
		01h	Двунаправленный параллельный порт
		02h	Параллельный порт совместимый с ECP 1.X
		03h	Контроллер IEEE 1284
		FEh	Целевое устройство IEEE 1284 (не контроллер)
	02h	00h	Многопортовый последовательный контроллер

Таблица П2.9 *(окончание)*

Базовый класс	Подкласс	Интерфейс	Значение
	03h	00h	Модем общего типа
		01h	Hayes-совместимый модем, 16450-совместимый интерфейс (см. далее)
		02h	Hayes-совместимый модем, 16550-совместимый интерфейс (см. далее)
		03h	Hayes-совместимый модем, 16650-совместимый интерфейс (см. далее)
		04h	Hayes-совместимый модем, 16750-совместимый интерфейс (см. далее)
	04h	00h	Контроллер GPIB (IEEE 488.1/2)
	05h	00h	Смарт-карта
	80h	00h	Другое коммуникационное устройство

Для Hayes-совместимых модемов первый регистр базового адреса (по смещению 10h) отображает соответствующее подмножество регистров (т. е. 16450, 16550 и т. д.) для последовательного контроллера от начала отображенного пространства. Эти регистры могут быть отображены либо в память, либо в пространство ввода-вывода в зависимости от типа регистра базового адреса.

Базовый класс 08h

Данный базовый класс определен для всех типов групповой системной периферии. Определены несколько подклассов, большинство из которых имеют специфический хорошо известный программный интерфейс на уровне регистров. Табл. П2.10 содержит все возможные значения для данного класса.

Таблица П2.10. *Значения базового класса 8*

Базовый класс	Подкласс	Интерфейс	Значение
08h	00h	00h	Контроллер PIC 8259 общего типа
		01h	Контроллер PIC ISA
		02h	Контроллер PIC EISA

Таблица П2.10 *(окончание)*

Базовый класс	Подкласс	Интерфейс	Значение
		10h	Контроллер прерываний ввода-вывода APIC (см. далее)
		20h	Контроллер прерываний I/O(x) APIC
	01h	00h	Контроллер DMA 8237 общего типа
		01h	Контроллер DMA ISA
		02h	Контроллер DMA EISA
	02h	00h	Системный таймер 8254 общего типа
		01h	Системный таймер ISA
		02h	Системные таймеры EISA (два таймера)
	03h	00h	Контроллер RTC общего типа
		01 h	Контроллер RTC ISA
	04h	00h	Контроллер PCI Hot-Plug общего типа
	80h	00h	Другая системная периферия

Для контроллера прерываний ввода-вывода APIC регистр базового адреса по смещению 0x10 используется для запроса минимум 32 байтов памяти, не доступной для упреждающего чтения. Два регистра внутри этого пространства размещены по смещению Base+0x00 (I/O Select Register) и Base+0x10 (I/O Window Register). Полное описание правил использования этих регистров приведено в документации 82420/82430 PCIset EISA Bridge Databook #290483-003.

Базовый класс 09h

Данный базовый класс определен для всех типов входных устройств (устройств ввода). Имеется несколько подклассов. Для контроллеров игрового порта определен программный интерфейс на уровне регистров. Табл. П2.11 содержит все возможные значения для данного класса.

Таблица П2.11. *Значения базового класса 9*

Базовый класс	Подкласс	Интерфейс	Значение
09h	00h	00h	Контроллер клавиатуры
	01h	00h	Цифровое перо

Таблица П2.11 *(окончание)*

Базовый класс	Подкласс	Интерфейс	Значение
	02h	00h	Контроллер мыши
	03h	00h	Контроллер сканера
	04h	00h	Контроллер игрового порта общего типа
		10h	Контроллер игрового порта
	80h	00h	Контроллер ввода другого типа

Базовый класс 0Ah

Данный базовый класс определен для всех типов базовых блоков. Программный интерфейс на уровне регистров не определен. Табл. П2.12 содержит все возможные значения для данного класса.

Таблица П2.12. *Значения базового класса A*

Базовый класс	Подкласс	Интерфейс	Значение
0Ah	00h	00h	Базовый блок общего типа
	80h	00h	Базовый блок другого типа

Базовый класс 0Bh

Данный базовый класс определен для всех типов процессоров. Имеется несколько значений подклассов, они соответствуют различным типам процессоров или набору инструкций. Программный интерфейс на уровне регистров не определен. Табл. П2.13 содержит все возможные значения для данного класса.

Таблица П2.13. *Значения базового класса B*

Базовый класс	Подкласс	Интерфейс	Значение
0Bh	00h	00h	386
	01h	00h	486

Таблица П2.13 *(окончание)*

Базовый класс	Подкласс	Интерфейс	Значение
	02h	00h	Pentium
	10h	00h	Alpha
	20h	00h	PowerPC
	30h	00h	MIPS
	40h	00h	Сопроцессор

Базовый класс 0Ch

Данный базовый класс определен для всех типов контроллеров последовательных шин. Имеются несколько значений подклассов. Определен программный интерфейс на уровне регистров для контроллеров USB и IEEE 1394. Табл. П2.14 содержит все возможные значения для данного класса.

Таблица П2.14. *Значения базового класса C*

Базовый класс	Подкласс	Интерфейс	Значение
0Ch	00	00h	IEEE 1394 (FireWire)
		10h	IEEE 1394 в соответствии со спецификацией 1394 OpenHCI
	01h	00h	Шина ACCESS
	02h	00h	SSA
	03h	00h	Универсальная последовательная шина (USB) в соответствии со спецификацией Universal Host Controller
		10h	Универсальная последовательная шина (USB) в соответствии со спецификацией Open Host Controller
		20h	Главный контроллер USB2 в соответствии с интерфейсом Intel Enhanced Host Controller
		80h	Универсальная последовательная шина без специфического программного интерфейса
		FEh	USB-устройство (не главный контроллер)

Таблица П2.14 *(окончание)*

Базовый класс	Подкласс	Интерфейс	Значение
	04h	00h	Волоконно-оптический канал
	05h	00h	SMBus
	06h	00h	InfiniBand
	07h (см. далее *примечание 1*)	00h	Интерфейс IPMI SMIC
		01h	Интерфейс IPMI Kybd Controller Style
		02h	Интерфейс IPMI Block Transfer
	08h (см. далее *примечание 2*)	00h	Стандарт интерфейса SERCOS (IEC 61491)
	09h	00h	Шина CANbus

Примечания

1. Определения интерфейса регистров для интерфейса IPMI (Intelligent Platform Management Interface) (подкласс 07h) приведено в спецификации IPMI.
2. Здесь отсутствует определение уровня регистров для стандарта SERCOS Interface standard. Для получения информации см. IEC 61491.

Базовый класс 0Dh

Данный базовый класс определен для всех типов контроллеров беспроводной связи. Имеются несколько значений подклассов. Программный интерфейс на уровне регистров не определен. Табл. П2.15 содержит все возможные значения для данного класса.

Таблица П2.15. *Значения базового класса D*

Базовый класс	Подкласс	Интерфейс	Значение
0Dh	00	00h	iRDA-совместимый контроллер
	01h	00h	Контроллер абонента IR
	10h	00h	RF-контроллер
	11h	00h	Bluetooth

Таблица П2.15 *(окончание)*

Базовый класс	Подкласс	Интерфейс	Значение
	12h	00h	Broadband
	80h	00h	Другой тип контроллера беспроводной связи

Базовый класс 0Eh

Данный базовый класс определен для всех типов контроллеров интеллектуального ввода-вывода. Основные характеристики этого базового класса проявляются в том, что функция ввода-вывода обеспечивает следующие несколько видов групповых определений для контроллеров ввода-вывода. Табл. П2.16 содержит все возможные значения для данного класса.

Таблица П2.16. *Значения базового класса E*

Базовый класс	Подкласс	Интерфейс	Значение
0Eh	00	xxh	Стандарт Intelligent I/O (I2O) Architecture Specification 1.0
		00h	FIFO-сообщение по смещению 040h

Спецификация для интеллектуальной архитектуры ввода-вывода (Intelligent I/O) может быть скачана с адреса: **ftp.intel.com/pub/IAL/i2o/**.

Базовый класс 0Fh

Этот базовый класс определен для контроллеров спутниковых коммуникаций, используемых для спутниковой связи. Табл. П2.17 содержит все возможные значения для данного класса.

Таблица П2.17. *Значения базового класса F*

Базовый класс	Подкласс	Интерфейс	Значение
0Fh	01h	00h	TV
	02h	00h	Звук
	03h	00h	Голос
	04h	00h	Данные

Базовый класс 10h

Данный базовый класс определен для всех типов криптоконтроллеров. Имеются несколько значений подклассов. Программный интерфейс на уровне регистров не определен. Табл. П2.18 содержит все возможные значения для данного класса.

Таблица П2.18. *Значения базового класса 10*

Базовый класс	Подкласс	Интерфейс	Значение
10h	00h	00h	Сетевой и вычислительный криптоконтроллер
	10h	00h	Демонстрационный криптоконтроллер
	80h	00h	Криптоконтроллер другого типа

Базовый класс 11h

Данный базовый класс определен для всех типов контроллеров захвата данных и обработки сигналов. Имеются несколько значений подклассов. Программный интерфейс на уровне регистров не определен. Табл. П2.19 содержит все возможные значения для данного класса.

Таблица П2.19. *Значения базового класса 11*

Базовый класс	Подкласс	Интерфейс	Значение
11h	00h	00h	DPIO-модули
	01h	00h	Счетчики производительности
	10h	00h	Синхронизация коммуникаций и тестирование/измерение частотных и временных параметров
	20h	00h	Плата управления
	80h	00h	Контроллер захвата данных/обработки сигналов другого типа

ПРИЛОЖЕНИЕ 3

Информация VPD

Vital Product Data (VPD) — это информация, которая уникально идентифицирует аппаратное обеспечение и потенциально — элементы программного обеспечения системы. Использование VPD позволяет обеспечить систему информацией типа: номер версии, серийный номер и другая детальная информация об устройстве. С системной точки зрения назначение VPD состоит в предоставлении этой информации владельцу системы и обслуживающему персоналу. Поддержка VPD является необязательной.

Физически VPD располагается в устройстве хранения (например, последовательный EEPROM) в PCI-устройстве. Доступ к VPD обеспечивается путем использования списка функциональностей в конфигурационном пространстве. Структура VPD-функциональности имеет следующий формат (рис. П3.1) [9].

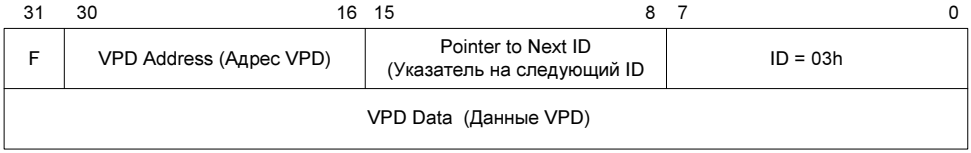


Рис. П3.1. Структура VPD Capability

Поля регистра имеют следующее назначение:

- **ID** — структура Capability ID 03h, поле доступно только для чтения;
- **Pointer to Next ID** — указатель на следующую структуру функциональности, значение "00h" соответствует последней структуре в списке. Поле доступно только для чтения;
- **VPD Address** — адрес информации VPD, к которой будет осуществлен доступ. Адрес выровнен на границу двойного слова. Регистр доступен для

чтения/записи, и начальное значение по включению питания не определено;

- **F** — флаг индикации, что передача данных между регистром VPD Data и устройством хранения выполнена. Запись в регистр флага производится, когда в регистр VPD Address записано какое-либо значение. Для чтения VPD-информации при записи в регистр VPD Address во флаг записывается "0". Аппаратная часть устройства установит флаг в "1", когда 4 байта данных будут переданы из устройства хранения в регистр VPD Data. Программное обеспечение может отслеживать флаг и, после установки его в "1", читать VPD-данные из регистра VPD Data. Если в один из регистров VPD Address или VPD Data была произведена запись до установки бита флага в "1", то результат первоначальной операции чтения может быть непредсказуем. Чтобы записать VPD-данные или прочитать/записать часть VPD-пространства, сначала выполняется запись в регистр VPD Data. Затем в регистр VPD Address записывается адрес, по которому будут сохранены VPD-данные, и бит флага устанавливается в "1" (во время, когда записывается адрес). Затем ПО отслеживает бит флага, и когда он установлен в "0" (аппаратной частью устройства), VPD-данные (все 4 байта) были переданы из регистра VPD Data в устройство хранения. Если в один из регистров VPD Address или VPD Data была произведена запись до установки бита флага в ноль, то результат записи в устройство хранения непредсказуем;
- **VPD Data** — данные VPD могут быть прочитаны через этот регистр. Последний значимый байт этого регистра (по смещению 4 в этой структуре функциональности) соответствует байту VPD по адресу, определяемому регистром VPD Address. Чтение или запись данных в этот регистр производится путем обычной PCI-передачи данных. Между этим регистром и устройством хранения VPD всегда передаются четыре байта. Чтение или запись данных вне VPD-пространства не допускается. Поле "VPD Data" (оба типа полей доступны для чтения и для чтения/записи) предназначено для хранения информации, а не для управления какими-либо операциями устройства. Начальное значение этого регистра по включению не определено.

Каждая плата расширения может содержать VPD. Если плата расширения стандарта PCI содержит несколько устройств, то информация VPD требуется только в одном из них, но может быть включена в каждое. PCI-устройства, разработанные для использования только на системной плате, могут также поддерживать необязательные (дополнительные) VPD-регистры.

VPD в PCI-плате расширения использует два предопределенных имени дескриптора, предварительно определенные в спецификации "Plug and Play ISA Specification", и две новых единицы, определенные специально для PCI VPD.

Используются следующие имена дескрипторов PnP ISA: "Identifier String" (0x02) для типа данных "Large Resource" и "End Tag" (0xF) для типа данных "Small Resource". Новые названия пунктов "large resource" для VPD являются VPD-R со значением данных только для чтения 0x10 и VPD-W со значением для чтения/записи данных 0x11.

Структура представляемой информации VPD разработана в соответствии с типами данных "Small" и "Large Resource", как описано в спецификации "Plug and Play ISA Specification", версия 1.0a. Использование этих структур данных минимизирует количество дополнительных ресурсов, необходимых для поддержки. Этот формат данных состоит из серий (последовательностей) структур данных. Типы данных из спецификации "Plug and Play ISA Specification", версия 1.0a, воспроизведены далее в табл. П3.1 и П3.2.

Таблица П3.1. *Определение битов дескриптора типа данных "Small Resource"*

Смещение в байтах	Имя поля
0	Значение = 0xxxxyyyb (Тип = Small(0), имя пункта Small = xxxx, длина = yy байт)
от 1 до n	Актуальная информация

Таблица П3.2. *Определение битов дескриптора типа данных "Large Resource"*

Смещение в байтах	Имя поля
0	Значение = 1xxxxxxxb (Тип = Large(1), имя пункта Large = xxxxxxx)
l	Длина пунктов данных, биты [7:0] (lsb)
2	Длина пунктов данных, биты [15:8] (msb)
От 3 до n	Актуальные пункты данных

Дескриптор "Identifier String" (0x02) является первым VPD и обеспечивает имя продукта устройства. Один дескриптор VPD-R (0x10) используется как заголовок для ключевых слов, доступных только для чтения, и один дескриптор VPD-W (0x11) используется как заголовок для ключевых слов, доступных для чтения/записи. Для списка VPD-R (включая тег и длину) должна быть вычислена контрольная сумма. Устройство хранения, содержащее данные для чтения/записи, является энергонезависимым устройством, которое будет хранить данные при отключении питания. Попытка записи области, доступной только для чтения, будет выполнена как пустой операнд. Последний де-

скриптор носит название "End Tag" (0x0F). На рис. П3.3 представлен пример дескрипторов типов данных типичной информации VPD.

Таблица П3.3. *Флаги типов данных для типичной VPD*

Смещение дескриптора, байт	Наименование дескриптора
0x2	Строка идентификации
0x10	Список VPD-R, содержащий одно или больше ключевых слов VPD
0x11	Список VPD-W, содержащий одно или больше ключевых слов VPD
0xF	Последний дескриптор "End Tag"

Формат VPD

Информационные поля внутри типа ресурсов VPD состоят из 3-байтного заголовка, располагающегося после некоторого количества данных (табл. П3.4). 3-байтный заголовок содержит 2-байтное ключевое слово и 1-байтный размер.

Keyword (ключевое слово) это двухсимвольная (ASCII) мнемоника, которая уникально идентифицирует данные в поле. Последний байт заголовка представляет значение длины данных в байтах.

Таблица П3.4. *Формат VPD*

Keyword		Length	Data
Байт 0	Байт 1	Байт 2	Байты 3 до n

Ключевое слово VPD состоит в списках в двух категориях: поля, доступные только для чтения и доступные для чтения/записи. Если не оговорено специально, поля данных ключевых слов представляются как ASCII-символы. Использование ASCII позволяет данным ключевого слова передаваться через различные вычислительные структуры без различий перевода. В качестве примера VPD пункта "серийный номер платы расширения" может быть приведен следующий:

- Keyword: SN S в байте 0 (см. табл. П3.4) и N в байте 1.
- Length: 08h
- Data: "00000194"

Совместимость VPD

Необязательные VPD поддерживались в предыдущей версии спецификации PCI. Для информации о предыдущем определении VPD необходимо обратиться к версии 2.1 спецификации "PCI Local Bus Specification".

Определение VPD

Далее приведены существующие дескрипторы VPD large и small ресурсов данных и ключевые слова VPD. Список может быть расширен в любое время, компании, желающие определить новые ключевые слова, должны связаться с PCI SIG. Все неопределенные значения являются зарезервированными для назначения SIG.

Дескрипторы VPD

VPD содержится в четырех типах дескрипторов данных large и small. В PCI-устройствах могут быть обеспечены следующие теги и поля ключевых слов VPD.

Таблица П3.5. *Дескрипторы данных VPD*

Тип дескриптора	Описание дескриптора
Large resource type Identifier String Tag (0x2)	Тег является первым пунктом в устройстве хранения VPD. Он содержит имя платы расширения в символьном виде
Large resource type VPD-R Tag (0x10)	Тег содержит доступные только для чтения ключевые слова VPD для платы расширения
Large resource type VPD-W Tag (0x11)	Тег содержит доступные для чтения/записи ключевые слова VPD для платы расширения
Small resource type End Tag (0xF)	Тег идентифицирует конец VPD в устройстве хранения

Доступные для чтения поля

В табл. П3.6 перечислены поля VPD, которые не могут быть изменены, так как содержат идентификационную информацию об устройстве.

Таблица П3.6. *Доступные для чтения поля*

Наименование поля	Расшифровка поля	Описание поля
PN	Add-in Card Part Number	Ключевое слово является расширением к Device ID (или Subsystem ID), определенному в заголовке конфигурационного пространства
EC	EC Level of the Add-in Card	Уровень технических изменений для этой платы расширения, хранится в символьном виде
FG	Fabric Geography	Ключевое слово обеспечивает стандартный интерфейс для инвентаризации структуры устройств на плате CompactPCI и применимо для нескольких коммутируемых структур. Данное ключевое слово определено и используется в спецификации поддерживаемой группой PCI Industrial Computer Manufacturers Group (PICMG) и доступно на **www.picmg.org**. Его основное предназначение состоит в использовании только в разработках на основе спецификаций PICMG
LC	Location	Ключевое слово обеспечивает стандартный интерфейс для определения размещения платы CompactPCI. Для примера, данное поле может содержать номер физического слота и шасси или номер стойки, где установлена плата. Данное ключевое слово определено и используется в спецификации поддерживаемой группой PICMG и доступно на **www.picmg.org**. Его основное предназначение состоит в использовании только в разработках на основе спецификаций группы PICMG
MN	Manufacture ID	Обеспечивает расширение к Vendor ID (или Subsystem Vendor ID), содержащемуся в заголовке конфигурационного пространства. Данное ключевое слово предоставляет производителям большую гибкость для дополнительного уровня идентификации деталей с целью установления происхождения данного устройства
PG	PCI Geography	Обеспечивает стандартный интерфейс для определения географии слота PCI (отражение между номерами физических слотов и логических адресов PCI) сегмента периферийных слотов, созданных платой CompactPCI. Данное ключевое слово определено и используется в спецификации поддерживаемой группой PICMG и доступно на **www.picmg.org**. Его основное предназначение состоит в использовании только в разработках на основе спецификаций группы PICMG

Таблица П3.6 *(окончание)*

Наименование поля	Расшифровка поля	Описание поля
SN	Serial Number	Уникальный серийный номер платы расширения, хранится в символьном виде
Vx	Vendor Specific	Содержимое пункта определяется производителем, хранится в символьном виде. Второй символ (x) ключевого слова может принимать значения от 0 до Z
CP	Extended Capability	Данное поле обеспечивает идентификацию новых функциональностей в области VPD. Данные этого поля идентифицируют, в каком адресном пространстве — памяти или ввода-вывода — располагаются регистры контроля/статуса для данной функциональности. Размещение регистров контроля/статуса указывается значением индекса (величина между 0 и 5) в регистре базового адреса. Это значение определяет адресный диапазон, который содержит регистры, и смещение внутри диапазона регистров базовых адресов, где размещены регистры контроля/статуса. Размер данного поля составляет четыре байта. Первый байт содержит ID дополнительной функциональности. Второй байт содержит индекс (основание ноль) используемого регистра базового адреса. Следующие два байта содержат смещение (в порядке little endian) внутри этого адресного диапазона и определяют регистры контроля/статуса для данной функциональности
RV	Checksum and Reserved	Первый байт данного пункта является байтом контрольной суммы. Контрольная сумма вычисляется, чтобы сумма всех байтов в VPD (от адреса VPD0 и до этого байта включительно) равнялась нулю. Остаток этого пункта является зарезервированным пространством (как необходимый) для идентификации последнего байта, доступного только для чтения пространства. Область чтения/записи не имеет контрольной суммы. Поле обязательно для реализации

Доступные для чтения/записи поля

Поля, перечисленные в табл. П3.7, специально предназначены для записи в них различной изменяемой информации, их значения могут быть модифицированы.

Таблица П3.7. *Доступные для чтения/записи поля*

Наименование поля	Расшифровка поля	Описание поля
Vx	Vendor Specific	Пункт, определяемый системой. Хранится в символьном виде. Второй символ (x) может принимать значения от 0 до Z
Yx	System Specific	Пункт, определяемый системой. Хранится в символьном виде. Второй символ (x) может принимать значения от 0 до 9 и от B до Z
YA	Asset Tag Identifier	Пункт, определяемый системой. Хранится в символьном виде. Содержит идентификатор системы, обеспечиваемый ее владельцем
RW	Remaining Read/Write Area	Этот дескриптор используется для идентификации неиспользуемой части пространства, доступного для чтения/записи. Производитель устройства инициализирует этот параметр на основании размера пространства для чтения/записи или пространства оставшегося после пункта Vx VPD. Необходимы один или более пунктов Vx, Yx и RW

Пример VPD

В качестве примера в табл. П3.8 приведена структура VPD, заполненная произвольным образом.

Таблица П3.8. *Пример типичной VPD*

Смещение, байт, дес.	Пункт	Значение
0	Large Resource Type ID String Tag (0x02)	0x82 "Product Name"
1	Длина	0x0021
3	Данные	"ABCD Super-Fast Widget Controller"
36	Large Resource Type VPD-R Tag (0x10)	0x90
37	Длина	0x0059
39	VPD Keyword	"PN"

Таблица П3.8 *(окончание)*

Смещение, байт, дес.	Пункт	Значение
41	Длина	0x08
42	Данные	"6181682A"
50	VPD Keyword	"EC"
52	Длина	0x0A
53	Данные	"4950262536"
63	VPD Keyword	"SN"
65	Длина	0x08
66	Данные	"00000194"
74	VPD Keyword	"MN"
76	Длина	0x04
77	Данные	"1037"
81	VPD Keyword	"RV"
83	Длина	0x2C
84	Данные	Checksum
85	Данные	Reserved (0x00)
128	Large Resource Type VPD-W Tag (0x11)	0x91
129	Длина	0x007C
131	VPD Keyword	"V1"
133	Длина	0x05
134	Данные	"65A01"
139	VPD Keyword	"Y1"
141	Длина	0x0D
142	Данные	"Error Code 26"
155	VPD Keyword	"RW"
157	Длина	0x61
158	Данные	Reserved (0x00)
255	Small Resource Type End Tag (0xF)	0x78

Список литературы

1. CompactPCI Specification Short Form Revision 2.1, 1997.
2. CompactPCI. Новейший международный стандарт промышленных и коммуникационных компьютеров — контроллеров на основе шины PCI. RTSoft.
3. Mini PCI Specification Revision 1.0, 1999.
4. PCI BIOS SPECIFICATION Revision 2.1, 1998.
5. PCI Bridge/Memory Controller TSPC106, Atmel, 2002.
6. PCI Bus Power PCI Bus Power Management Interface Specification Revision 1.1, 1998.
7. PCI Express Base Specification Revision 1.0, 2002.
8. PCI Hot-Plug Specification Revision 1.0, 1998.
9. PCI Local Bus Specification, Revision 2.3, 2002.
10. PCI Local Bus Specification, Revision 2.2, 1998.
11. PCI Local Bus Specification Production Version, Revision 2.1, 1995.
12. PCI Mobile Design Guide Version 1.1, 1998.
13. PCI-to-PCI Bridge Architecture Specification Revision 1.1, 1998.
14. PCI-X Addendum to the PCI Local Bus Specification Revision 1.0a, 2000.
15. Small PCI Specification Version 1.5a Final, 1998.
16. System Management Bus (SMBus) Specification Version 2.0, 2000.
17. Борзенко А. PCI: жизнь продолжается. PCWeek/RE, № 47, 2000.
18. Володин А. Ю., Горбачев С. В., Шейкин Ю. Е. Шина PCI в высокопроизводительных микропроцессорных системах: учебное пособие. — СПб.: СПбГУАП, 1999.

19. Гольдштейн Л. Д., Зернов Н. В. Электромагнитные поля и волны. — М.: Советское радио, 1971.
20. Гук М. Аппаратные средства IBM PC. — СПб.: Питер, 2002.
21. Зернов Н. В., Карпов В. Г. Теория радиотехнических цепей. — Л.: Энергия, 1972.
22. Уинн Л. Библия по техническому обеспечению Уинна Роша: Пер. с англ. А. Пашковского. — Минск: МХХК "Динамо", 1992.
23. Хоровиц П. Хилл У. Исскуство схемотехники: Пер. с англ. — Изд. 6-е. — М.: Мир, 2003.
24. Шевкопляс Б. В. Микропроцессорные структуры. Инженерные решения. — М.: Радио и связь, 1990.
25. Якусевич В. В. BIOS Setup. Полное руководство. — М.: Альтекс-А, 2004.

Предметный указатель

P

R

S

T

U

V

А

Б

В

Г

Д

Е

З

И

К

Л

М

Н

О

П

P

C

T

У

Ф

Ц

Ч

Ш

Э