Четыре скорости UEFI BIOS

Михаил Закусило

Парадокс эволюции постоянных запоминающих устройств состоит в том, что все технические новации направлены на превращение ROM в RAM.Таким новшеством стало электрическое стирание, а затем и технология флеш-памяти, с успехом применяющаяся в чипах, хранящих код BIOS. Всё бы хорошо, но существенным недостатком использования ПЗУ была и остается их низкая производительность. Ее помогает обойти использование «теневой памяти» (Shadow RAM) в которую для ускорения доступа копируется BIOS (а теперь и UEFI). Почему бы не попытаться выполнить старт персональной платформы, полностью отказавшись от использования оперативной памяти? Давайте рассмотрим этот вопрос на примере реализации микрокомпьютера Intel Compute Stick.

После того, как мы заглянули внутрь стик-компьютера, совершенно очевидно, что обзор возможностей современных реализаций флеш-памяти стоит начать с анализа чипа W25Q64FV, используемого в Compute Stick для хранения кода UEFI BIOS. Компания Winbond, разработавшая этот чип, позиционирует его как устройство, способное выполнять программы непосредственно из исходного носителя. Данная технология получила название Execute In Place (XIP) и по идее должна заменить режим Shadow RAM.

Рис 1. *Микросхема флеш-памяти W25Q64FV на плате Intel Compute Stick*

В широком смысле технология Execute In Place – это выполнение программы непосредственно из исходного носителя (в нашем случае в качестве исходного носителя используется микросхема ROM) без копирования кода в оперативную память. По сути, современный термин XIP эквивалентен олдскульному подходу, известному еще со времен первых персональных компьютеров до эпохи «теневой памяти. В современных системах с момента старта платформы до инициализации ОЗУ (либо до инициализации режима Cache-as-RAM), код UEFI BIOS также выполняется в режиме Execute In Place. Другими словами, особой новизны здесь нет.

Разумеется, для универсальной микросхемы ROM не имеет значения, какая информация читается из нее: выполняемый код, данные, либо идет побайтное копирование образа ROM в оперативную память. Поэтому к инициативам Winbond можно относиться скептически, и неверно утверждать, что есть ROM с поддержкой XIP, а есть ROM без поддержки XIP. Любой ROM поддерживает XIP, поэтому правильнее говорить о том, что некоторые ROM (в частности рассматриваемая микросхема W25Q64FV) ***оптимизированы для ускорения XIP***.

Имеется в виду следующее. Производительность памяти, в том числе постоянных запоминающих устройств, характеризуется двумя параметрами: пропускной способностью (количество данных, прочитанных за единицу времени при чтении непрерывного блока) и латентностью (время реакции на изменившийся адрес).

Очевидно, при **копировании ROM в Shadow**, имеет место чтение большого непрерывного блока. В этом случае важнее пропускная способность. Опережающее чтение байтов, расположенных в окрестности текущего читаемого байта, приносит пользу. Время передачи адреса в этом случае не существенно, так как он передается единожды для большого непрерывного блока, а затем автоинкрементируется внутри микросхемы ROM во время последовательного продвижения по блоку.

При выполнении программного кода (**это и есть XIP**), наоборот, имеют место ветвления (условные переходы), выборочный доступ к мелким фрагментам, частое изменение адреса по нерегулярному закону. В этом случае важнее минимизировать латентность, а опережающее чтение байтов, явно не затребованных для чтения, может принести вред, так как мы с высокой вероятностью потратим время работы SPI интерфейса на чтение данных, которые не потребуются, например, прочитаем инструкции, расположенные в программе после инструкции передачи управления JMP.

Смысл XIP-оптимизированных режимов чтения в том, чтобы обеспечить низкую латентность (это важно при чтении мелко разбросанных данных), а также «умерить инициативу» микросхемы ROM и контроллера SPI по чтению данных, которые не затребованы явно.

В документе W25Q64FV Data Sheet принято некоторое отступление от теоретических канонов и маркетинговое упрощение, согласно которому любой быстрый ROM можно называть XIP-оптимизированным, просто потому, что если Shadow не используется, то производительность ROM в большей степени влияет на производительность устройства. Поэтому XIP-оптимизацией в документе названы и такие оптимизации, которые повлияют не только на доступ в режиме XIP, но и на непрерывный блоковый доступ, имеющий место при копировании содержимого ROM в Shadow RAM. В частности расширения SPI-протокола: Dual SPI, Quad SPI.

Режим Dual SPI увеличивает разрядность передаваемых данных от одного бита (классической реализации) до двух. Согласно документации, функциональность контактов микросхемы флеш-памяти переопределяется следующим образом:

- DI (Data Input, контакт 5) = IO0
- DO (Data Output, контакт 2) = IO1

В результате линии, образующие однобитовую двунаправленную шину, становятся шиной двухбитовой. ***Режим Quad SPI*** увеличивает разрядность передаваемых данных до четырех. Согласно документации, функциональность контактов микросхемы переопределяется следующим образом:

- DI (Data Input, контакт 5) = IO0
- DO (Data Output, контакт 2) = IO1
- WP (Write Protect, контакт 3) = IO2
- HOLD (State hold, контакт 7) = IO3

Линии, образующие однобитовую двунаправленную шину и сигналы управления, переопределяются с использованием мультиплексирования в четырехбитовую шину. А вот ***режим QPI (Quad Peripheral Interconnect)*** в чипе W25Q64FV по праву можно назвать XIP-оптимизацией. Он использует тот же набор сигналов, что и Quad SPI, описанный выше. Отличие в том, что в режиме QPI, увеличение разрядности шины в 4 раза затрагивает не только передачу адреса и данных, но и управляющих команд: 8-битная команда передается не за восемь, а за два такта. В результате снижается латентность (минимизируется потеря времени на восприятие новой команды). Как было показано выше, латентность особенно важна для XIP при доступе к мелко-разбросанным данным.

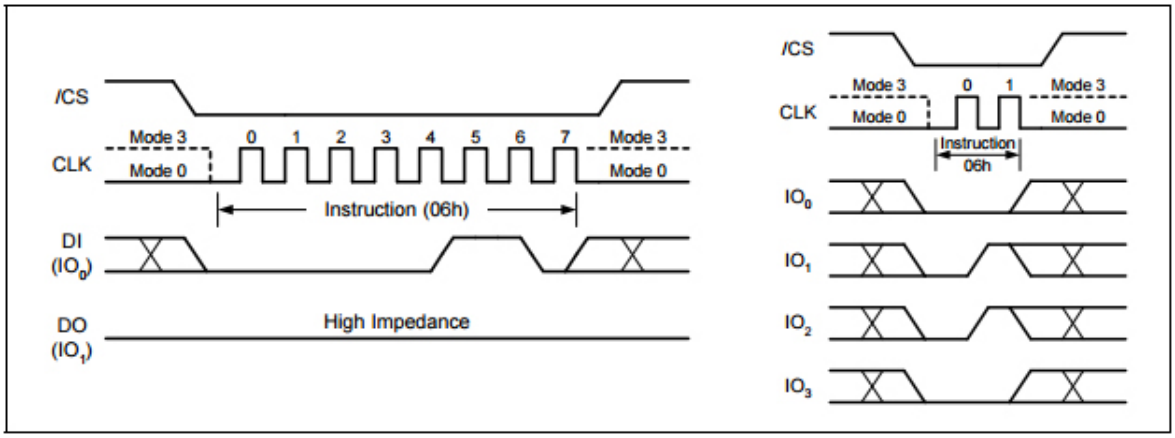

Рис 2. *Write Enable в режиме SPI (слева) и QPI (справа), реализованные в W25Q64FV*

Интересный факт: для декларирования функциональных различий микросхема изменяет код идентификатора устройства (Device ID), если включен режим QPI. В режиме SPI, код Device ID = 4017h, а в режиме QPI, он становится равным 6017h.

MANUFACTURER ID	**(MF7 - MF0)**	
Winbond Serial Flash	EFh	
Device ID	**(ID7 - ID0)**	**(ID15 - ID0)**
Instruction	**ABh, 90h, 92h, 94h**	**9Fh**
W25Q64FV (SPI)	16h	4017h
W25Q64FV (QPI)	16h	6017h

Рис 3. *Идентификаторы флеш-памяти W25Q64FV в зависимости от используемого интерфейса*

Строго говоря, если называть последовательным (serial) интерфейсом только такой интерфейс, у которого для передачи данных используется одна линия, то SPI с указанными расширениями уже не является последовательным. Пропускная способность возрастает пропорционально увеличенной разрядности. Использование описанных функциональных расширений возможно только в том случае, если SPI хост-контроллер их поддерживает.

Обычно, хост-контроллер входит в состав микросхемы PCH (Platform Controller Hub), либо находится в составе центрального процессора. В нашем случае, когда мы анализируем архитектуру Intel Compute Stick, имеет место второй вариант. В документе «*Intel Atom Processor Z3600 and Z3700 Series Data Sheet*» упоминается только классический протокол SPI. Скорее всего, разработчики процессора посчитали этот аспект не актуальным, ориентируясь на системы с Shadow RAM, в которых режим XIP имеет место только в момент старта Boot-блока.

И еще одна деталь – поскольку рассматриваемые функциональные расширения являются программно-управляемыми, желательно чтобы поддержка расширенной функциональности носителя firmware была реализована в составе процедур UEFI BIOS. Хотя, теоретически, это ограничение обойти можно.

Вместо послесловия

Не следует путать QPI (*Quad Peripheral Interface*) и QPI (*Quick Path Interconnect*). Во втором случае – это шина межпроцессорной связи в Intel Xeon и не имеет никакого отношения к обсуждаемой теме.