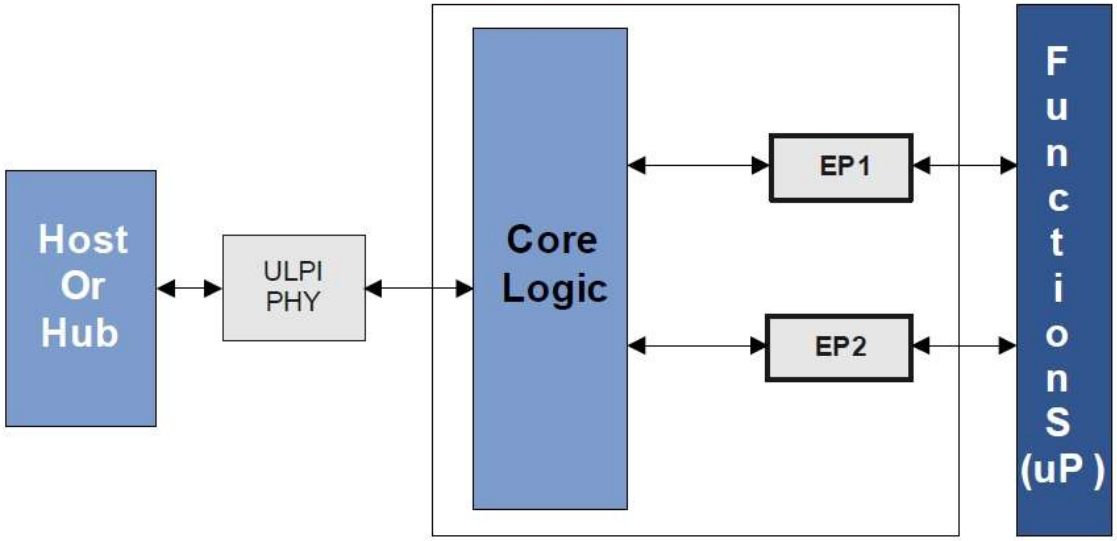
3. Функциональное описание

Это ядро было разработано специально для интерфейса Avalon. На рис. 3-1 кратко показано, как ядро подключается к функции микроконтроллера и хоста.

Функционирование IP ядра контролируется функцией микроконтроллера посредством регистров конечных точек.

Figure 3-1. Operational Block Diagram



По умолчанию USB ядро использует внутри чиповую память (512 x 32) для всех буферов конечных точек IN и 512 x 32) для всех буферов конечных точек OUT. В зависимости от количества доступной памяти внутри чипа, размер памяти может быть увеличен в SOPC Builder для улучшения скоростных характеристик модуля. Обратитесь к проекту в начале руководства за подробной информацией о параметрах IP ядра. Механизм двойной буферизации используется для уменьшения задержек, необходимых программе, и увеличивает производительность USB.

Конечные точки

Ядро USB поддерживает 3 конечные точки (Control, Bulk IN, Bulk OUT). Функция микроконтроллера должна задать конечные точки (кроме конечной точки Control) записью в регистр конечных точек: EPn_CSR, EPn_INT, EPn_BUFx.

Буфер указателей

Буфер указателей указывает на структуру данных входа/ выхода в памяти. Если буферы не определены, ядро может ответить в USB хост квитированием NAK.

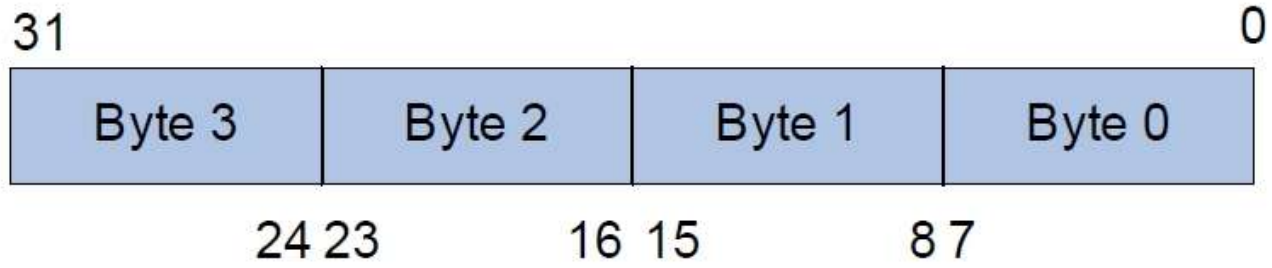
Это ядро использует средство двойной буферизации, которое уменьшает задержку, необходимую функции микроконтроллера и программному драйверу. Данные могут быть извлечены/ помещены из/ в буферы в виде круговой системы. Когда данные перемещаются в/ из конечной точки, используется первый буфер buffer0. Когда первый буфер опустошён/ заполнен, функция микроконтроллера может быть извещена прерыванием. Функция микроконтроллера должна снова заполнить/ опустошить buffer0, а ядро будет использовать buffer1 для следующей операции. Когда второй буфер заполнен/ опустошён, вызывается прерывание для микроконтроллера, а ядро снова использует buffer0, и так далее. Буфер, который использует набор битов, вынуждает ядро отвечать хосту квитированием NAK/ NYET.

Буфер, использующий биты, отображается, когда был использован этот буфер (эта информация также присутствует в регистре источника прерывания). Функция микроконтроллера должна сбрасывать эти биты после того, как она опустошит/ заполнит буфер.

Организация данных

Так как размер памяти буфера имеет 32 битную глубину, а USB определяет транзакции в границах байта, важно понимать зависимость распределения данных в буфере для восстановления правильной последовательности USB байтов. Это ядро USB поддерживает порядок от младшего к старшему.

Figure 3-2. Data Organization



Указатель буфера всегда указывает на байт 0. Ядро USB всегда забирает четыре байта из памяти буфера. Последний байт, который передаётся в транзакцию Nth, зависит от размера буфера. Максимальный размер пакета должен быть кратным 4 байтам.

Прерывания

Ядро USB имеет выход прерываний. Выход полностью программируемый. Использование прерываний предназначено для системы, в которую включено USB ядро.

Прерывания в ядре USB имеют два уровня иерархии:

1. **Основной регистр источника прерываний** (INT_SRC) отображает прерывания, не зависимые от конечной точки. Эти прерывания отображают все события, имеющие глобальное значение для всех конечных точек, или наоборот, не связанные с конечной точкой из-за состояния ошибки.
2. **Регистр источника прерываний конечной точки** отображает события, относящиеся к конечной точке.

Временные параметры

Выход прерывания устанавливается по возникновению события, разрешённого в маске регистра прерывания. Он остаётся установленным, пока читается основной регистр прерываний.

Поведение программы

Обработчик прерывания должен сначала прочитать основной регистр источника прерываний (INT_SRC), чтобы определить источник прерываний. Он должен запомнить значение, которое нужно прочитать, пока не будут обработаны все источники прерываний. Если установлен любой из битов 2 или 1, обработчик прерываний должен прочитать также соответствующий регистр прерываний конечной точки, чтобы определить событие, относящееся к этой конечной точке. В любое время могут отображаться значения от нескольких источников прерываний. Программа может быть подготовлена для обработки каждого источника прерываний надлежащим образом.

Необходимо позаботиться о том, чтобы не потерять источники прерываний, после того, как основной регистр источников прерываний сбросится после чтения.

4. Подключение шины Avalon

Шина Avalon - это простейшая архитектура шины, разработанная для подключения программного процессора и периферии внутри SOPC системы. Шина Avalon - это интерфейс, задающий подключение портов между компонентами мастер и слейв, и определяющий временных параметры взаимосвязи этих компонентов.

Подключение по шине Avalon показывает, каким образом система сопрягается с шиной Avalon, где один компонент - мастер, один - слейв, и как остальные устройства подключены к системе. В этой системе компонент IP ядро USB20HR работает как слейв. Это компонент, работающий по прерыванию. Интерфейс Avalon слейв используется для доступа к данным USB2.0. На рис. 4-1 показано подключение шины Avalon.

Figure 4-1. Avalon Bus Connectivity