Программируемая логическая матрица в микроконтроллере ADuC7025

Иосиф Каршенбойм, iosif.karshenboim@eltech.spb.ru

ВВЕДЕНИЕ

Полупроводниковые кристаллы дешевеют. Микроконтроллеры тоже дешевеют, но в меньшей степени. Просто дело в том, что они становятся более мощными и более высокоскоростными. Разрастается их периферия. Постепенно разрядность увеличивается с 8-ми бит до 32-х бит. Но становится ли микроконтроллер быстрее?

На первый взгляд все просто: выше тактовая частота – больше производительность процессора.

Однако, при внимательном рассмотрении выясняется, что не все определяется только тактовой частотой, а иногда и типом примененного ядра процессора. Есть еще и другие параметры, непосредственно влияющие на производительность микроконтроллера и всего канала управления, выполненного на этом микроконтроллере.

В этой статье будут представлены только отдельные фрагменты того, что влияет на производительность микроконтроллера, но и при этом еще и с учетом конкретного применения. Причем этот обзор выполнен только «под влиянием аппаратных средств». Специалисты по операционным системам могли бы продолжить и значительно дополнить эту дискуссию, но, конечно «со стороны софта».

И еще одно небольшое отступление. Недавно в конференциях была дискуссия о том, какие процессоры мы применяем. См. Л[1]. Начало сообщения здесь мы пропускаем, поскольку оно зависит от возраста корреспондента. Типичное сообщение выглядело так: “…PIC…, …AVR…, собираюсь перейти на ARM”. Поэтому в данной статье хочется более детально рассмотреть вопрос быстродействия именно в приложении к ARM.

Контроллер с ядром ARM – это уже довольно сложная архитектура, которая может включать в себя не только собственно микроконтроллер, но и различные дополнительные узлы, такие как каналы DMA и контроллеры динамической памяти. Поэтому быстродействие всего устройства в целом будет определяться не только производительностью ядра микроконтроллера, но и его способностью реагировать на события, происходящие во “внешнем мире”. Реакция на внешние события возможна, только в том случае, если ресурсы необходимые для этой реакции свободны и находятся в распоряжении микроконтроллера. Например, регенерация динамической памяти закончена и есть возможность доступа к данным, находящимся в этой памяти. Или завершена пересылка массива данных в порт, и шина освобождается для микроконтроллера. Но, поскольку данные узлы являются дополнением к той основе, которая называется ARM, то мы не будем здесь их рассматривать.

ЧЕМ МИКРОКОНТРОЛЛЕР СВЯЗАН С «ВНЕШНИМ МИРОМ»?

Да и здесь ответ давно известен: линии связи, порты ввода-вывода и входы контроллера прерываний.

По поводу линий связи обсуждать нечего, тонны бумаги и миллионы букв потрачены на это обсуждение. Опрос портов – тут тоже все тривиально. Получил прерывание от часов, проверил расписание обхода и опросил.

Далее сам контроллер прерываний. Здесь необходимо учитывать латентность (задержку на выполнение) контроллера. Контроллер – статический автомат, шифрующий состояния приоритетов. На входе у него регистр маски прерываний, триггер глобального разрешения-запрета. На прохождение сигнала через эти цепи требуется некоторое время. Далее, сигнал запроса приходит в АЛУ и …. Но перед этим должна завершиться текущая выполняемая команда. И эта команда не всегда бывает однотактная.

В качестве примера возьмем ADuC7025. В самом худшем случае время ожидания для FIQ состоит из самого большого отрезка времени, которое может потребоваться для сигнала запроса, чтобы пройти через синхронизатор, плюс время для завершения самой длинной команды – LDM, которая загружает все регистры, включая PC, плюс время, необходимое для аварийного прекращения работы данных, плюс время для входа в FIQ . В конце этого времени, ARM7TDMI будет выполнять команду, расположенную по адресу 0x1C (адрес вектора прерывания FIQ). Максимальное время – 50 циклов процессора, которое равно 1,2 мкс для системы, использующей для процессора синхрочастоту в 41,78 МГц. Максимальное вычисление времени ожидания запроса на прерывание IRQ подобно приведенному выше, но необходимо учесть тот факт, что FIQ имеет более высокий приоритет и может задержать вход в запрос на прерывание IRQ, поскольку обслуживание прерывания верхнего уровня может проводиться в течение произвольного отрезка времени.

Минимальное время ожидания может быть уменьшено до 42 циклов, если не используется команда LDM , поэтому некоторые компиляторы имеют такую опцию, позволяющую компилировать, не используя данную команду. Другая опция, позволяющая уменьшить латентность до 22 циклов, состоит в том, что программа должна выполниться в режиме THUMB.

Минимальное время ожидания для FIQ или прерываний IRQ – всего пять циклов, которое состоит из времени, необходимого для того, чтобы запрос мог пройти через синхронизатор плюс время, требуемое для выполнения режима исключения.

СЛОЖНЫЙ «ВНЕШНИЙ МИР»

К сожалению, «внешний мир», находящийся вокруг микроконтроллера с каждым годом становится все больше и сложнее. Что можно было получить от микроконтроллера с 1 К памяти? Пара прерываний и все. Главный цикл программы с вызовом подпрограмм обслуживания прерываний. Теперь, поскольку микроконтроллеры стали значительно мощнее, требований, а значит и задач, выполняемых ими, стало значительно больше. Больше внешних воздействий приходит на микроконтроллер. Значит ли это, что число запросов прерываний стало больше? С одной стороны – да. Входов запросов прерываний действительно стало больше. Но как это число связано с количеством обслуживаемых задач? Известно, что в любом микроконтроллере есть два самых больших узла, если судить по объему занимаемого ими оборудования. Это дешифратор команды и шифратор приоритета в контроллере прерываний. Поэтому невозможно значительно увеличить число запросов на прерывание и при этом получить быстрый шифратор.

Здесь автор позволит себе несколько отклониться от обсуждения контроллера прерываний.

Хочется подчеркнуть тенденцию, имеющую место в развитии микроконтроллеров. С одной стороны мы видим микроконтроллеры с абсолютно “фиксированным” “железом”. С другой стороны – софт-микроконтроллеры, загружаемые в FPGA. Посредине – различные “гибридные” системы на кристалле, у которых часть функций фиксирована при изготовлении кристалла, и часть функций может быть реконфигурирована пользователем. Среди таких систем на кристалле могут быть как конфигурируемые пользователем, так и изготовляемые под заказ. Что здесь является основным критерием для того, чтобы получить минимальное время реакции? Для микроконтроллеров с “фиксированным” “железом” задача оптимизации выполнена разработчиком микросхем. Пользователь имеет только стандартные возможности по управлению прерываниями. Если ядро софт-микроконтроллера закриптовано, то пользователь такого микроконтроллера находится в том же положении, что и пользователь микроконтроллера с “фиксированным” “железом”. Но вот в том случае, когда пользователь имеет возможность переконфигурировать узел контроля прерываний, есть возможность сократить число линий запросов прерываний, уменьшить объем шифратора прерываний, а, следовательно, несколько повысить быстродействие всего устройства в целом. Контекстное переключение банков регистров – понятие известное и привычное. А вот контекстное переключение входов запросов прерываний чем хуже? Или контекстное переключение флагов и битов порта? А запуск АЦП в зависимости от контекста и бита порта? Именно таким путем и пошла фирма ADI, разрабатывая свои микросхемы серии ADuC. Весь микроконтроллер выполнен как чип с “фиксированным” “железом”, кроме одного узла, а именно узла конфигурации запросов на прерывание. Кроме собственно запросов на прерывания могут конфигурироваться еще ряд сигналов, среди которых сигнал запуска АЦП.

Рассмотрим пример реализации узла обработки запросов на прерывание, на примере микросхемы ADuC7025, см. Л[2]. Основное свойство этого узла – возможность конфигурации и реконфигурации. Конфигурация и реконфигурация производится под управлением микроконтроллера, как при старте системы, так и в процессе работы. Чем же так хороша эта технология? А тем, что в зависимости от выполняемой задачи, на входы запросов прерываний могут быть подключены разные источники воздействия. При этом можно изменить не только сам источник запроса, но и дисциплину его обслуживания. Узел, выполняющий данные функции у микроконтроллеров ADuC7025 называется PLA (Programmable Logic Array – программируемая логическая матрица). Правда, если говорить более строго, то PLA представляет собой массив программируемой логики и набор шин и мультиплексоров. Блок-схема узла PLA приведена на рис. 1.

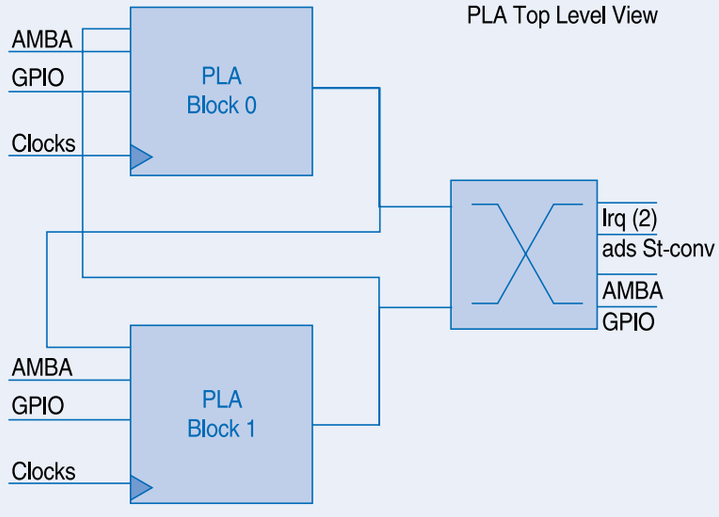

Рис. 1. Блок-схема узла PLA

Узел PLA состоит из трех частей. Две части – два одинаковых PLA-блока, каждый из которых представляет собой массив PLA, они отличаются между собой только внешними подключениями. На входы этих массивов поступают сигналы от портов ввода-вывода, от процессорной шины и сигналы тактирования. Третий блок – коммутирует выходные сигналы на входы запросов прерываний, на порты ввода-вывода и на процессорную шину, а так же и на сигнал запуска АЦП.

Каждый массив PLA содержит входной мультиплексор тактовой частоты и 8 одинаковых элементов, отличающихся подключениями входных и выходных цепей. Схема одного из этих элементов массива приведена на рис. 2. Элемент массива PLA состоит из входных мультиплексоров, узла перекодировки двух переменных, выходного триггера и выходного мультиплексора. Входные мультиплексоры выбирают два входных сигнала из набора сигналов, поступающих на входы элемента. Узел перекодировки выполняет одну из следующих функций, см. табл.1.

Таблица 1 Функции, выполняемые узлом перекодировки

№	Функция	№	Функция
1	0	9	AND
2	NOR	10	EXNOR
3	A AND NOT B	11	B
4	NOT A	12	NOT A OR B
5	NOT A AND B	13	A
6	NOT B	15	A OR NOT B
7	EXOR	15	OR
8	NAND	16	1

Пользователь имеет возможность подать сигнал запроса на триггер и зафиксировать запрос прерывания под нужную ему частоту, или пропустить сигнал запроса в обход триггера и обслуживать запрос по уровню, таким как он есть.

Далее, с выхода элементов PLA сигналы поступают на выходные мультиплексоры и с их выхода – на запросы прерываний IRQ0 и IRQ1. Выходные сигналы элементов также поступают и на мультиплексор, выбирающий сигнал управления для запуска АЦП. Также выходные сигналы элементов можно подать на порты микросхемы. Для этого необходимо загрузить соответствующий код в регистры, разрешающие прохождение сигналов на порты микросхемы.

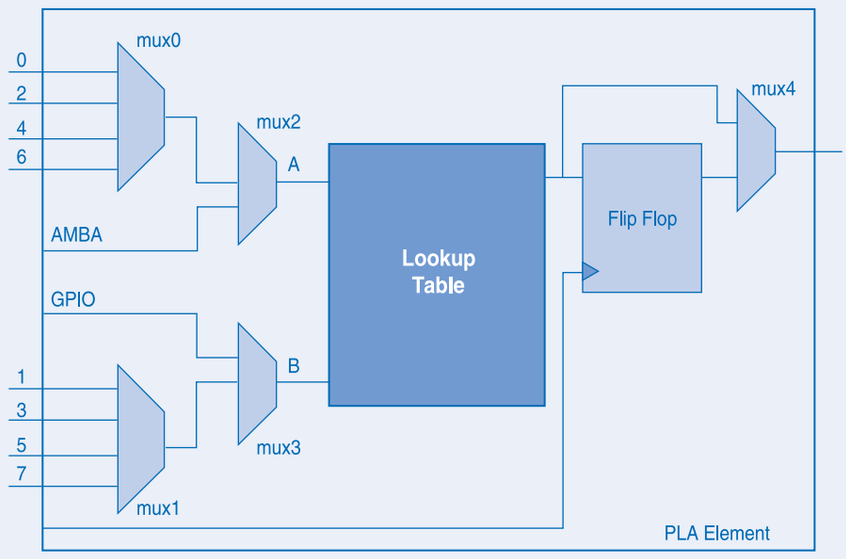

Рис. 2. Блок-схема элемента PLA

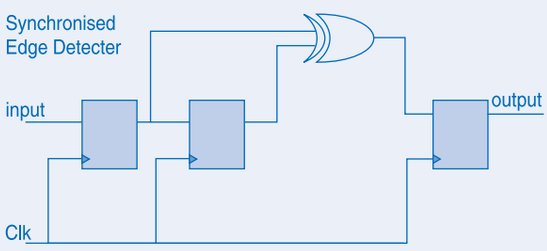

Рис. 3. Схема формирователя сигналов запроса прерывания по фронту входного сигнала

Программный инструмент для конфигурации PLA – ADuC PLA Tool.

Фирма-изготовитель микросхем предоставляет программный инструмент для генерации кодов, позволяющих настроить требуемые режимы работы PLA. Выходными данными этой программы будут коды микроконтроллера, написанные в ассемблере или на Си. Настройка режимов работы узлов PLA производится в графическом режиме.

В качестве примера конфигурации рассмотрим схему, приведенную на рис. 3. В этом примере выходные воздействия формируются при переходах входного сигнала из состояния «0» в состояние «1».

Данный пример конфигурации выполняется при помощи программы ADuC PLA Tool так, как показано на рис. 4.

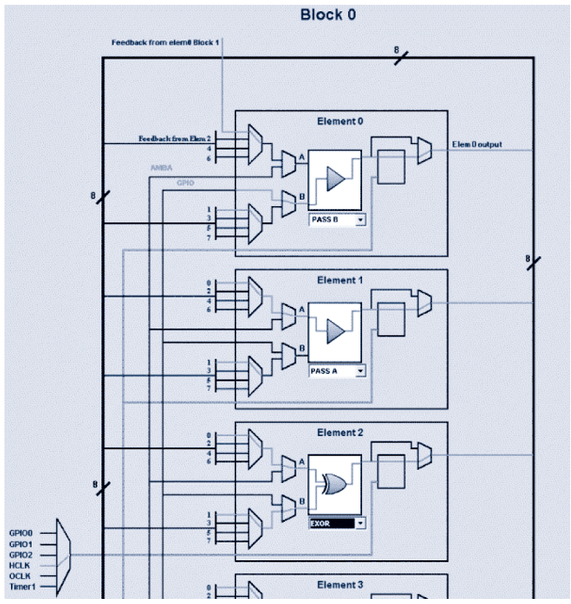

Рис. 4. Установка настроек в программе ADuC PLA Tool

К программе ADuC PLA Tool дается подробное описание всех пунктов конфигурации узла PLA. Результатом работы программы является фрагмент кодов, выполненный на языке Си или в ассемблере. Пример программы инициализации PLA, выполненный на языке Си, приведен на рис. 5.

```
/ Code Generated By the ADuC 702X PLA Tool
// FileType:              C PLA Configuration File
// Source: C Source Code
// Date:                  23.11.2005 11:32:08
//================================================
void        plaInitialize( )
{
// Configure Port Pins for PLA mode
// In order for the PLA Tool to configure the required GPIO pins
// you must make the necessary selections on the outputs tab!!!

        // Configure individual elements
        PLAELM0 = 0x000A;
        PLAELM1 = 0x001B;
        ….
        PLAELM7 = 0x001E;

// Clk Source configuration
        PLACLK = 0x0000;
}
```

Рис. 5. Пример программы инициализации PLA, выполненный на языке Си

ВЫВОДЫ

Конфигурация и реконфигурация позволяют облегчить обработку информации, увеличить быстродействие устройства, а, следовательно, повысить его потребительские качества.

Литература

1. http://forum.electronix.ru/index.php?showtopic=5&st=0
2. ADuC7024_25_PrD.pdf