

Содержание

Предисловие	12
Глава 1. RISC-архитектура для встроенных приложений.	13
1.1. Введение	13
1.2. RISC-подобное ядро C166S V2: предпосылки создания	13
1.3. Использование RISC-архитектуры во встроенных системах управления	16
1.3.1. Шинный интерфейс	17
1.3.2. Реакция на прерывание RISC-процессора	18
1.3.3. Регистры и многозадачность	18
1.3.4. Использование сокращенного набора команд RISC-процессора	22
1.4. Связь RISC-процессора с внешними устройствами	23
1.5. Преимущества использования RISC-процессоров во встроенных приложениях ...	24
1.6. Сравнение новых и традиционных RISC-процессоров	25
Глава 2. Начинаем работу с микроконтроллерами XC166.	28
2.1. Ключевые вопросы	28
2.1.1. Обзор семейства микроконтроллеров	28
2.1.2. Основные принципы проектирования	29
2.1.3. Установка опций аппаратной конфигурации микроконтроллера	29
2.2. Расчёт номиналов конфигурирующих резисторов	31
2.3. Стартовая конфигурация микроконтроллера	33
2.3.1. Конфигурация для внутреннего старта	33
2.3.2. Конфигурация для внешнего запуска	34
2.4. Управление сбросом	36
2.5. Тактовые сигналы и их источники	38
2.5.1. Запуск системы ФАПЧ (PLL)	39
2.5.2. Запуск от внешней шины	40

2.5.3. Внутренний старт от встроенного ПЗУ	40
2.5.4. Выбор тактовой частоты	41
2.5.5. Выбор значений регистра PLLCON	43
2.6. Генерация тактовых импульсов	45
2.6.1. Типы источников тактовых импульсов	45
2.6.2. Расчёт схемы кварцевого генератора	45
2.6.3. Процедура тестирования компонентов кварцевого генератора	47
2.6.4. Разводка цепей тактового генератора на печатной плате	49
2.6.5. Признаки неверного расчёта схемы тактового генератора	50
2.6.6. Тактовый генератор часов реального времени	51
2.6.7. Дополнительная информация по проектированию тактового генератора	51
Глава 3. Режимы работы и синхронизация шины	52
3.1. Гибкий шинный интерфейс	52
3.1.1. Использование линий выбора микросхем (CS)	52
3.1.2. Перекрытие сигналов выбора (CS)	54
3.2. Настройка режима работы шины	55
3.2.1. Внутрисхемная загрузка	55
3.2.2. Внешняя загрузка	55
3.3. Оптимизация системы адресации	55
3.3.1. Время доступа к внешней памяти	56
3.3.2. Расчёт временных соотношений для мультиплексированной шины	56
3.3.3. Расчёт временных соотношений для немultipлексированной шины	58
3.3.4. Программные средства для расчёта временных параметров синхронизации шины	60
Глава 4. Сопряжение с внешними устройствами памяти	62
4.1. Использование 16-битных микросхем памяти	63
4.2. Использование 8-битных микросхем памяти в 16-битных системах на базе XC166	65
4.3. Использование сигнала /VNE при работе с 8-битной памятью	67
4.4. Сопряжение микросхем DRAM с микроконтроллерами семейства XC166	68
4.5. Использование карт флэш-памяти совместно с XC166	70
4.5.1. Недорогая гигабайтная память	70
4.5.2. Использование карт Compact Flash для обновления программ микроконтроллеров XC166	70
4.5.3. Интерфейс для подключения карт SD/Multimedia Card	71
4.5.4. Интерфейс для подключения карт Compact Flash	72
4.5.5. Управление флэш-картами с большим объёмом памяти	73
4.5.6. Программный интерфейс приложения (API) файловой системы для встроенных программ на языке C	74
4.5.7. Ресурсы, необходимые для реализации файловой системы для микроконтроллера XC166	75
6 Глава 5. Встроенная программируемая флэш-память	77

5.1. Введение	77
5.2. Организация внутренней флэш-памяти	78
5.3. Обеспечение надёжности флэш-памяти	79
5.3.1. Динамическая коррекция ошибок	80
5.3.2. Долговечность флэш-памяти	80
5.3.3. Обеспечение работоспособности флэш-памяти в особых условиях	81
5.3.4. Прогнозирование будущих отказов флэш-памяти	83
5.4. Программаторы флэш-памяти.	84
5.4.1. Программирование флэш-памяти микроконтроллера XC166 при массовом производстве	84
5.4.2. Краткие сведения о программах начальной загрузки	86
5.4.3. Тестирование программаторов флэш-памяти на соответствие IEC61508 и другим стандартам	88
5.5. Отладка режима начальной загрузки	89
5.5.1. Проблемы, возникающие при отладке режима начальной загрузки в случае использования модуля JTAG	89
5.5.2. Отладка режима начальной загрузки с использованием внутрисхемного эмулятора	90
5.6. Аппаратные аспекты программирования внутрисхемной флэш-памяти.	90
5.7. Программирование флэш-памяти через интерфейс CAN	93
5.8. Программирование флэш-памяти через интерфейс SPI	94
5.9. Программирование флэш-памяти in-situ («на месте»)	95
Глава 6. Схема распределения памяти	96
6.1. Регистры-указатели страниц данных (Data Page Pointer — DPP)	96
6.1.1. Быстрый доступ к данным с использованием DPP	96
6.1.2. Доступ к большим объёмам данных	98
6.1.3. Влияние способа адресации данных на компиляторы C/C++	98
6.2. Выбор схемы распределения памяти и конфигурации внешней шины	99
6.2.1. Использование регистров-указателей страницы данных (DPP)	99
6.2.2. Использование памяти PSRAM	100
6.2.3. Влияние внешней шины на производительность	100
6.2.4. Внутренняя флэш-память	100
6.2.5. Внешнее ПЗУ (ROM)	100
6.3. Увеличение количества линий ввода/вывода	102
Глава 7. Оптимизация потребляемой мощности	104
7.1. Уменьшение потребляемой мощности путём оптимизации тактовой частоты	105
7.2. Сравнение токов потребления разных микроконтроллеров	106
7.3. Напряжение питания	106
Глава 8. Системное программирование	107
8.1. Передача данных через последовательные порты	107
8.1.1. Синхронные последовательные порты	107

Содержание

8.1.2. Модуль I ² C	108
8.2. Подключение USB-устройств к микроконтроллерам семейства XC166	109
8.2.1. Подключение микроконтроллера XC166 к шине USB с помощью микросхемы FT245B	110
8.2.2. Программное обеспечение для реализации USB-интерфейса	112
8.2.3. Начало работы с USB-интерфейсом	112
8.3. Обслуживание запросов на прерывание	113
8.3.1. Задержки при обработке прерываний	113
8.3.2. Программные прерывания	114
8.3.3. Аппаратные прерывания	114
8.3.4. Структура прерываний	115
8.3.5. Замечания по использованию системы прерываний	116
8.4. Пересылка данных с использованием периферийного контроллера событий (PEC)	117
8.5. Организация стеков в микроконтроллерах семейства XC166	119
8.6. DSP-сопроцессор для микроконтроллеров семейства XC166	120
8.6.1. Реализация функций DSP в микроконтроллерах	120
8.6.2. Работа компилятора с модулем MAC	121
8.6.3. DSP-библиотеки фирмы Infineon	121
8.7.1. Синхро-временной режим CAN, запускаемый с помощью модуля TwinCAN	123
8.7.2. Режим замкнутой передачи CAN	124
Глава 9. Назначение выводов и портов в пользовательском приложении	125
9.1. Общие замечания о параллельных портах ввода/вывода	125
9.2. Назначение линий портов микроконтроллера	126
9.2.1. Порт 0	126
9.2.2. Порт 1	127
9.2.3. Порт 2	127
9.2.4. Модули CAPCOM	128
9.2.5. Порт 3	138
9.2.6. Порт 4	143
9.2.7. Порт 5	145
9.2.8. Порт 6	146
9.2.9. Порт 7	146
9.2.10. Порт 9	147
9.2.11. Порт 20	147
9.3. Прерывания по линиям порта	148
Глава 10. АЦП микроконтроллера XC166	149
10.1. Расширенные режимы аналого-цифрового преобразования	150
10.2. Базовая частота преобразования АЦП	151
10.3. Калибровка АЦП	151
10.4. Защита аналоговых входов от повышенного напряжения	152

10.5. Согласование входов АЦП с источниками сигналов	153
10.6. Аналоговое опорное напряжение	157
10.7. Вопросы разработки печатной платы	157
10.7.1. Размещение компонентов	157
10.7.2. Источник питания	157
10.7.3. Шины заземления	158
10.8. Подключение АЦП к источникам сигналов	158
10.8.1. Метод измерения отношения	158
10.8.2. Источник опорного напряжения с фиксированной точностью	160
10.8.3. Режим с коррекцией преобразования	161
10.8.4. Измерение аналоговых напряжений, превышающих уровень 5 В	162
Глава 11. Типовые применения микроконтроллеров семейства XC166	163
11.1. Применение в автомобильной электронике	164
11.2. Применение в промышленных системах управления	165
11.3. Применение в системах телекоммуникации	166
11.4. Применение на транспорте	167
11.5. Применение в потребительской радиоэлектронике	167
11.6. Применение в измерительных приборах	168
11.7. Медицинское и аэрокосмическое оборудование	168
Глава 12. Совместимость XC166 с микроконтроллерами другой архитектуры . . .	170
Глава 13. Монтаж микроконтроллеров семейства XC166 на печатной плате	172
13.1. Типы корпусов	172
13.2. Подключение эмуляторов к микроконтроллерам семейства XC166	172
13.2.1. Типы разъёмов для корпусов микроконтроллеров	172
13.2.2. Отладка устройств на основе микроконтроллера семейства XC	173
13.3. Подключение внутрисхемного эмулятора	175
13.3.1. Схема QuadConnect	175
13.3.2. Разъём фирмы Yamaichi	177
13.3.3. Припаиваемая сборка	178
13.3.4. Припаиваемые эмулирующие микроконтроллеры	178
13.3.5. Адаптеры «PressOn» для эмулятора	179
13.4. Разводка печатных плат для микроконтроллеров семейства XC166	180
Глава 14. Включение и настройка новых плат	181
14.1. Оборудование для настройки	181
14.2. Прежде, чем включить питание	182
14.3. Тестирование платы	183
14.3.1. Приложения с внешним запуском	183
14.3.2. Использование режима последовательной загрузки и утилиты MINIMON для тестирования новых плат	185

Содержание

14.3.3.Использование JTAG для тестирования новых плат	187
14.3.4.Типичные проблемы при работе с внешней шиной	192
14.3.5.Тестирование устройств с внутренним запуском	192
14.3.6.Тестирование системы	192
Заключение	194
Дополнительная литература	195
Приложение 1. Система обозначений микроконтроллеров семейства XC166	196
Приложение 2. Цоколёвка основных модификаций микроконтроллеров XC166 .	197

БЛАГОДАРНОСТИ

*Авторы хотели бы поблагодарить
Карла Смита (Karl Smith),
Майка Коплэнда (Mike Copeland)
и Манфреда Чутку (Manfred Choutka)
из фирмы Infineon Technologies,
Хоакима Клейна (Joachim Klein)
из фирмы Hitex Development Tools GmbH,
а также Стива Брауна (Steve Brown)
и Венди Уокера (Wendy Walker)
и выразить свою благодарность
сотням пользователям
микроконтроллеров семейства 166
в Великобритании
за их вклад в написание этой книги.*

Предисловие

Данное руководство содержит основные сведения, необходимые для разработки вашего первого устройства на основе микроконтроллера XC166. Это простые, но основополагающие сведения, знание которых, особенно на начальном этапе разработки, позволит сэкономить немало времени и денег. Цель этой книги – дополнить официальное руководство пользователя XC166 теми фактами, которые пригодятся вам на практике.

Кое-что из материала книги можно отыскать в спецификациях на семейство XC166, но большая часть информации получена в результате нашей практической работы и найти ее можно только в данной книге. Многие из поднятых в ней тем отнюдь не очевидны или вообще не рассматривались ранее. Если руководство пользователя на микроконтроллер XC166 достаточно полно освещает какой-либо вопрос, то в книге делается ссылка на него и информация не дублируется. Данное издание является далеко не полным и немало дополнительной информации можно найти на веб-сайте фирмы Infineon.

Достоверность информации, представленной в данной книге, подтверждена специалистами фирмы Hitex. Однако, несмотря на это, фирма Hitex не берет на себя ответственности за возможные неточности. Те или иные субъективные или неподтвержденные сведения, содержащиеся в книге, не всегда отражают официальную точку зрения фирм Hitex Development Tools Ltd. и Infineon Technologies AG.

Материал подготовили:

Майкл Бич (Michael Beach)

Дэвид Гринхилл (David Greenhill)

Дополнительный материал предоставили:

Карл Смит (Karl Smith), Infineon Technologies UK

Хоаким Клейн (Joachim Klein), Hitex Development Tools

RISC-архитектура для встроенных приложений

1.1. Введение

Популярность RISC-микроконтроллеров непрерывно растёт. В процессорном ядре C166S V2, на базе которого построена серия микроконтроллеров XC166, широко используются принципы архитектуры вычислений с сокращённым набором команд (Reduced Instruction Set Computer — RISC), благодаря чему удалось достичь очень высокой производительности процессора при весьма умеренной его стоимости. Чтобы понять, почему именно RISC-технологии больше подходят для встроенных приложений, работающих в режиме реального времени, полезно будет вспомнить, как развитие традиционной архитектуры микропроцессоров с полным набором команд (Complex Instruction Set Computer — CISC), пик популярности которой пришёлся на период с конца 1980-х по середину 1990-х годов, привело к появлению RISC-архитектуры. Тем не менее, необходимо учитывать, что несмотря на множество достоинств, присущих RISC-технологии, не все RISC-микроконтроллеры, по определённым причинам, полностью их используют. В данной главе мы рассмотрим несколько таких «критических» случаев.

1.2. RISC-подобное ядро C166S V2: предпосылки создания

Причиной снижения интереса к микропроцессорам, использующим архитектуру с полным набором команд (CISC), стала необходимость существенно увеличить производительность микропроцессорных систем. В первую очередь, речь идёт о требованиях, предъявляемых графическими рабочими станциями и современными компьютерными видеоиграми. Традиционно, стандартные наборы ассемблерных команд (инструкций) микропроцессоров разрабатывались таким образом, чтобы максимально упростить работу программиста. Это достигалось путём широкого использования микрокодов, позволяющих создавать всё более мощные инструкции, такие, например, как команда перемножения трёх операндов.

дов. Благодаря этому, программист, пишущий на ассемблере (или разработчик компилятора языка высокого уровня), может добиваться тех же результатов, что и с применением простых команд, но с меньшей затратой сил.

Чем большее количество различных инструкций необходимо распознавать и обрабатывать (декодировать) центральному процессору, тем сложнее электронные схемы, размещённые на его кристалле, и тем большее количество циклов тактовой частоты занимает процесс обработки команд. Увеличение площади полупроводникового кристалла повышает стоимость устройства и потребляемую им мощность. Поскольку максимальная тактовая частота работы электронных схем микропроцессора физически ограничена возможностями полупроводниковых технологий, очевидно, что максимальная производительность напрямую зависит от количества машинных циклов, требуемых для выполнения команд процессора.

RISC-технология переносит основную нагрузку на программистов, пишущих на ассемблере и разработчиков компиляторов языка высокого уровня, а не на программирование в микрокодах. Исследования, проведённые учёными и производителями полупроводниковой техники, показали, что при одной и той же тактовой частоте корректно запрограммированный RISC-процессор обеспечивает гораздо более высокую производительность, чем процессор, использующий технологию CISC.

Как ни странно, разработчики встроенных приложений средней производительности не спешат отказываться от CISC-микроконтроллеров. Если в сфере высокопроизводительных устройств RISC-процессоры, такие как ARM9, MIPS и Hyperstone, создают жёсткую конкуренцию традиционным CISC-процессорам PowerPC и Pentium, то для более распространённых задач встроенных приложений RISC-устройства используются относительно редко. Между тем, возрастающая сложность современных алгоритмов управления встроенными устройствами приводит к тому, что выполнение практически любых задач, кроме самых элементарных, требует использования больших вычислительных мощностей. Кроме того, в таких областях, в отличие от использования микропроцессоров в рабочих станциях, весьма критична величина максимального времени реакции на недетерминированные события — параметр, с которым у CISC-процессоров дело обстоит особенно плохо. Впрочем, большинство RISC-процессоров ARM также не обладают выдающимися характеристиками по времени реакции.

Многие современные 16-битные микроконтроллеры средней производительности построены на базе существующих CISC-архитектур (S12, H8, M16C и т. п.), которые, как и 8-битные устройства (например, 8051), разработаны более 20 лет назад. Производители микросхем обеспечивают потребителям возможность модернизации существующих систем, поэтому микропроцессоры новых серий часто разрабатываются на базе уже существующих. Использование тех же или близких архитектуры и набора команд избавляет потребителя от необходимости тратить деньги на переписывание уже существующих ассемблерных программ.

Микроконтроллеры, как и рабочие станции, могут быть запрограммированы на языках высокого уровня, что позволяет сократить время написания программ и улучшить удобство обслуживания системы. К сожалению, использование даже

самых лучших компиляторов приводит к потерям быстродействия¹⁾, что ещё раз подчеркивает необходимость увеличения производительности процессоров.

Помимо прямой обработки данных, микроконтроллеры также должны получать информацию от периферийных устройств, таких как АЦП, ШИМ, таймеров, портов, ФАПЧ и т. п. Все это требует обработки информации в режиме реального времени и быстрой реакции на прерывания.

Ниже приведены основные недостатки, свойственные CISC-микроконтроллерам:

1. Длительное и непредсказуемое время ожидания прерывания

Во время выполнения сложных «снижающих трудозатраты программиста» ассемблерных команд все вычислительные ресурсы процессора оказываются задействованы, что не позволяет обслуживать поступающие извне прерывания. Это приводит к тому, что длительность ожидания обработки прерывания оказывается непредсказуемо велика, что может создать серьезные проблемы в быстродействующих системах реального времени. Одним из методов уменьшения времени реакции CISC-микроконтроллеров является интеграция в ядро «блока временной обработки» или использование внешнего процессора для разгрузки критичных по времени операций. Однако, это повышает сложность проектов и требует использования очень компактных микрокодов, в дополнение к куда более распространённому языку С и ассемблеру для программирования самого CISC-ядра.

2. Медленное декодирование вследствие слишком большого набора команд

Загруженные команды нужно распознать среди нескольких сотен или даже тысяч возможных, поэтому процесс декодирования замедляется и усложняется.

3. Замедление работы устройств памяти, вызываемое частым доступом к ним

Данные обычно берутся процессором из внешней памяти, и помещаются в регистры типа аккумулятора. С ними производятся математические или логические операции, затем результат снова записывается в память. Часто при выполнении процедуры то или иное значение может потребоваться несколько раз, что требует дополнительных обращений к памяти для записи или чтения.

4. Медленный вызов процедур

Если при вызове подпрограммы используются параметры (что чаще всего и происходит при качественном программировании на языке высокого уровня), то они по отдельности помещаются в стек. Затем эти параметры обрабатываются в регистре (регистрах) аккумулятора, и лишь после этого через стек возвращаются вызывающей подпрограмме.

5. Выполнение за один раз только одной операции

Каждое периферийное устройство или источник прерывания должны иметь специализированную обслуживающую подпрограмму, которая, как минимум, потребует помещения в стек и извлечения из него слова состояния программы и счётчика команд, а также передачи данных от периферийного устройства или к периферийному устройству.

¹⁾ По сравнению с программированием на языке ассемблера. (прим. ред.)

6. Пользовательская программа должна быть структурирована таким образом, чтобы соответствовать аппаратным возможностям микроконтроллера

Программирование встроенных приложений часто представляет собой множество отдельных задач, выполняемых в режиме реального времени и связанных в единую систему. В типовых CISC-процессорах переключение между задачами происходит довольно медленно. Часто при поступлении новой задачи оказывается, что многие регистры заняты и данные из них предварительно нужно поместить в стек. Эта проблема усугубляется при использовании компиляторов высокого уровня, характерным свойством которых является наличие в библиотечных функциях большого количества подлежащих сохранению локальных переменных.

7. Избыточные инструкции и режимы адресации

Почти повсеместное использование языков высокого уровня заставляет производителей микроконтроллеров обеспечивать совместимость с различными компиляторами путём введения в набор команд процессора соответствующих дополнительных инструкций. На практике, компиляторы чаще всего используют лишь несколько режимов адресации, а большое количество других инструкций остаются незадействованными, что усложняет процесс декодирования команд.

8. Несогласованные наборы команд процессора

Наборы команд становятся всё труднее использовать из-за большого количества доступных типов данных и разнообразия поддерживаемых режимов адресации.

9. Неполная загруженность шины

Во время выполнения сложных команд внутренняя шина микроконтроллера простаивает.

1.3. Использование RISC-архитектуры во встроенных системах управления

Чтобы продемонстрировать, как использование RISC-архитектуры позволяет увеличить производительность микроконтроллеров, в качестве примера, мы взяли процессорное ядро C166S V2.

Основные определения:

- 1 такт = $1/\text{частота генератора}$ — базовая единица времени, используемая при рассмотрении процессорных систем.
- 1 машинный цикл = такт — минимальное время, необходимое процессору для выполнения наиболее простой операции. Поскольку машинный цикл позволяет абстрагироваться от тактовой частоты, это понятие можно использовать для сравнения RISC- и CISC-процессоров.
- Все отчёты машинных циклов как для S12X, так и для C166 M2 приводятся в однокристалльном режиме работы.

1.3.1. Шинный интерфейс

Характерной особенностью RISC-процессоров является высокий уровень их конвейеризации, что позволяет увеличить скорость выполнения команд. За счет перекрытия различных стадий, в любом машинном цикле могут одновременно выполняться до 4 команд. Упрощённо эти стадии можно представить следующим образом:

FETCH	— выборка кода операции из памяти программ
DECODE	— идентификация кода операции из небольшого списка и полученных операндов выборки
EXECUTE	— выполнение операции, соответствующей найденному коду
WRITE-BACK	— результат выполнения операции возвращается в память по заданному адресу

Таким образом, хотя инструкция требует четырёх машинных циклов, реально она выполняется всего за один машинный цикл (1 такт). Конвейеризация значительно ускоряет выполнение последовательного (линейного) кода, поскольку шина гарантированно будет полностью загружена.

Ряд наиболее современных RISC-процессоров (например, C166S V2) имеют 5-ступенчатый конвейер исполнения команд (добавлена стадия ADDRESS), а также 2-ступенчатый блок выборки команд с упреждением PREFETCH:

Высокопроизводительный блок выборки команд (IFU)

PREFETCH	— выборка команд из памяти в предсказанном порядке; обнаруживаются любые ветвления, их выполнение или невыполнение обуславливаются логикой предсказания
FETCH	— адрес следующей команды для выборки вычисляется с использованием правил предсказания ветвления

Конвейерный блок

DECODE	— команды декодируются и, если требуется, производится доступ к регистровому файлу для считывания GPR (регистра общего назначения), используемого в режиме косвенной адресации
ADDRESS	— вычисляются все адреса операндов
MEMORY	— из памяти RAM и регистров извлекаются все необходимые операнды
EXECUTE	— выполняется операция, соответствующая коду
WRITE BACK	— результат операции возвращается в память по заданному адресу

Такая организация интерфейса теоретически обеспечивает удвоение производительности при выполнении линейного участка программы. Однако, даже при наличии блока предсказания ветвлений, который минимизирует снижение производительности, вызванное имеющимися в реальной программе ветвлениями, удвоения производительности достигнуть не удаётся.