МИНИСТЕРСВТО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ФЕРГАНСКОГО ФИЛИАЛА ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Факультет “Компьютерный инжиниринг”

Кафедра “Компьютерные системы”

КУРСОВАЯ РАБОТА

по предмету “*Технология настойки микропроцессорных систем*”

На тему: ПРОЕКТИРОВАНИЕ МИКРОПРОЦЕССОРНЫХ СИСТЕМ. УРОВНИ ПРЕДСТАВЛЕНИЯ МИКРОПРОЦЕССОРНОЙ СИСТЕМЫ

Подготовил: ***студент группы 642-11 Хасанов Р.***

Принял: ***Ахунджанов У.***

Фергана 2015г.

ОГЛАВЛЕНИЕ

1. Введение

Микропроцессорная система может быть описана на различных уровнях абстрактного представления.

Существующую микропроцессорную систему можно описать на любом известном уровне представления, но в начальной стадии проектирования ее можно описать только на концептуальном уровне. В процессе разработки системы происходит переход от одного уровня ее представления к другому, более детальному. Каждая абстракция несет в себе только информацию, которая соответствует данному уровню, и не содержит каких-либо сведений относительно более низких уровней.

Логический уровень присущ исключительно дискретным системам. На этом уровне выделяются два подуровня: переключательных схем и регистровых пересылок. Подуровень переключательных схем образуется вентилями и построенными на их основе операторами обработки данных. Переключательные схемы подразделяются на комбинационные и последовательностные; первые в отличие от последних не содержат запоминающих элементов. Поведение системы на этом уровне описывается алгеброй логики, моделью конечного автомата, входными/выходными последовательностями 1 и 0. Комбинационные схемы представляются таблицей истинности, в которой каждому входному набору значений сигналов ставится в соответствие набор значений сигналов на выходах. Последовательностные схемы могут описываться диаграммами или таблицами входов/выходов, в которых определены взаимно однозначные соответствия между входами схемы, внутренними состояниями (комбинациями значений элементов памяти) и выходами. Подуровень регистровых пересылок характеризуется более высокой степенью абстрагирования и представляет собой описание регистров и передачу данных между ними. Он включает в себя две части: информационную и управляющую. Информационная часть образуется регистрами, операторами

и путями передачи данных. Управляющая часть определяет зависящие от времени сигналы, инициирующие пересылку данных между регистрами.

Схемный уровень образуется резисторами и конденсаторами. Показателями поведения системы на этом уровне служат напряжение и ток, представляемые в функции времени или частоты. Этот уровень описания дискретной системы широко используется в описаниях аналоговых систем и не является ни наинизшим из возможных, ни достаточным для полной характеристики системы.

2. Основная часть

Проектирование микропроцессорных систем

О правильности функционирования микропроцессорной системы на уровне "черного ящика" с полностью неизвестной внутренней структурой можно говорить лишь тогда, когда произведены ее испытания, в ходе которых реализованы все возможные комбинации входных воздействий, и в каждом случае проверена корректность ответных реакций. Однако исчерпывающее тестирование имеет практический смысл лишь для простейших элементов систем. Следствием этого является тот факт, что ошибки проектирования встречаются при эксплуатации, и для достаточно сложных систем нельзя утверждать об их отсутствии на любой стадии жизни системы. В основе почти всех методов испытаний лежит та или иная гипотетическая модель неисправностей, первоисточником которой служат неисправности, встречающиеся в практике. В соответствии с мо-делью в рамках каждого метода предпринимаются попытки создания тестовых наборов, которые могли бы обеспечить удовлетворительное выявление моделируемых неисправностей. Любой метод тестирования хорош ровно настолько, насколько правильна лежащая в его основе модель неисправности.

Важным моментом является правильный выбор соотношения между степенью общности модели, стоимостью и степенью сложности

формирования и прогона тестов, ориентированных на моделируемые неисправности. Чем конкретнее модель, тем легче создать для нее систему тестов, но тем выше вероятность того, что неисправность останется незамеченной. Если же модель неисправностей излишне общая, то из-за комбинаторного возрастания числа необходимых тестовых наборов и/или времени вычислений, требуемого для работы алгоритмов формирования тестов, она станет непрактичной и пригодной только для несложных систем.

Ошибки, неисправности, дефекты

В жизненном цикле микропроцессорной системы, как любой дискретной системы, выделяются три стадии: проектирование, изготовление и эксплуатация. Каждая из стадий подразделяется на несколько фаз, для которых существуют вероятности возникновения конструктивных или физических неисправностей, приводящих систему в неработоспособное состояние. Поэтому на каждой фазе необходимы процедуры тестового контроля, направленные на обнаружение и локализацию неисправностей. Процедура тестового контроля может быть определена как проведение экспериментов с "черным ящиком". Дискретная система любой сложности или часть такой системы может рассматриваться как "черный ящик" с множеством входов и выходов. Правильность функционирования этого "черного ящика" должна устанавливаться путем подачи входных сигналов и наблюдения ответных выходных сигналов системы. В тех случаях, когда поведение "черного ящика" отличается от нормального, характеризуемого его спецификацией или представлениями человека, говорят о наличии ошибки. Ошибка вызывается некоторой неисправностью, представляющей собой некорректное состояние внутри "черного ящика". Неисправности классифицируют в соответствии с их причинами: физическая, если причиной ее служат либо дефекты элементов, либо физическое воздействие окружающей среды; субъективная (внесенная, нефизическая), если ее

причиной служат ошибки проектирования, неправильный монтаж элементов, грубые ошибки оператора. Физические неисправности - непредусмотренные, нежелательные изменения значения одной или нескольких логических переменных в системе. Субъективные неисправности - конкретные проявления недостатков программного и аппаратного обеспечения и неправильных действий оператора, имеющих место при выполнении дискретной системой предписанных спецификацией действий .

Под субъективными неисправностями подразумеваются неисправности нефизические, вызванные недостатками различных схем, конструкций, программ, средств эксплуатации - компиляторов, ассемблеров, программ автоматизации проектирования, инструкции по эксплуатации, процедур и средств контроля и т.д. Субъективные неисправности делят на проектные и интерактивные.

Проектные неисправности вызваны недостатками, вносимыми в систему на различных стадиях реализации исходного задания при структурном проектировании, разработке алгоритмов, написании программ, трансляции в машинный код, детальном логическом и техническом проектировании, а также при последующих модификациях аппаратного и программного обеспечения. Интерактивные неисправности возникают, когда в процессе работы, технического обслуживания или отработки системы оператор вводит в нее через интерфейс человек-машина ложную информацию, не соответствующую текущему состоянию системы. Как правило, это происходит в результате непонимания инструкции для оператора или вследствие неточностей ввода информации.

Ошибка - проявление неисправности (физической или субъективной). В зависимости от уровня иерархической структуры системы термин "ошибка" может иметь различный смысл. Так, для дискретного устройства он означает появление неверных двоичных сигналов ("0" вместо "1" и "1"

вместо "0"); для программы ошибка означает отклонение поведения программы от заданного, приводящее к выдаче неверных результатов.

Следует четко разграничивать понятия "ошибка" и "неисправность". Неисправность может приводить или не приводить к ошибке в зависимости от состояния системы. В то же время возникновение ошибки обязательно говорит о существовании какой-то неисправности. Одна и та же ошибка может быть вызвана множеством неисправностей, а одна неисправность может служить причиной целого ряда ошибок. Например, если триггер, предназначенный для хранения кода переполнения разрядной сетки ЭВМ, вследствие неисправности все время находится в состоянии "0", то ошибки из-за неисправности не будет до тех пор, пока в процессе вычислений реально не возникнет арифметическое перепол-нение, при котором триггер останется в состоянии "0" вместо перехода в состояние "1". Однако даже и в этом случае такая ошибка процессора не обязательно приведет к ошибке на программном уровне, если в программе условие переполнения не проверяется и, следовательно, ни с какой стороны не влияет на ее дальнейшее поведение.

Дефекты - физические изменения параметров компонентов системы, выходящие за допустимые пределы. Их называют сбоями, если они носят временный характер, и отказами, если они постоянны.

Существует такая причинноследственная связь:
1) дефект, представляющий собой изменение в значениях параметров компонентов, вызывает неисправность, т.е. отклонение от заданного значения (значений) логической переменной (переменных) в точке дефекта;
2) неисправность приводит к подаче неверных логических значений на вход (входы) остальной части системы и может вызывать ошибки, проявляющиеся при последующей работе других исправных логических схем;
3) ошибки приводят к появлению неправильных результатов или к отклонению от нормального хода исполнения программы.

Обнаружение ошибки и диагностика неисправности

Дефект не может быть обнаружен до тех пор, пока не будут созданы условия для возникновения из-за него неисправности, результат которой должен быть, в свою очередь, передан на выход испытуемого объекта, для того чтобы сделать неисправность наблюдаемой. Метод испытаний должен позволить генерировать тесты, ставящие испытуемый объект в условия, при которых моделируемые неисправности проявляли бы себя в виде обнаруживаемых ошибок. Если испытуемый объект предназначен для эксплуатации, то при обнаружении ошибки необходимо произвести локализацию неисправности с целью ее устранения путем ремонта или усовершенствования испытуемого объекта.

Диагностика неисправности - процесс определения причины появления ошибки по результатам тестирования. Отладка - процесс обнаружения ошибок и определение источников их появления по результатам тестирования при проектировании микропроцессорных систем. Средствами отладки являются приборы, комплексы и программы .

Точность, с которой тот или иной тест локализует неисправности, называется его разрешающей способностью. Требуемая разрешающая способность определяется конкретными целями испытаний. Например, при испытаниях аппаратуры в процессе эксплуатации для ее ремонта часто необходимо установить, в каком сменном блоке изделия имеется неисправность. В заводских условиях желательно осуществлять диагностику неисправности вплоть до уровня наименьшего заменяемого элемента, чтобы минимизировать стоимость ремонта. В лабораторных условиях в процессе отладки опытного образца необходимо определять природу неисправности (физического или нефизического происхождения). В случае возникновения и проявления дефекта требуется локализовать место неисправности с точностью до заменяемого элемента, а при проявлении субъективной неисправности - с точностью до уровня

представления (программного, схемного, логического и т. д.), на котором была внесена неисправность, и места.

Так как процесс проектирования микропроцессорной системы содержит неформализуемые этапы, то отладка системы предполагает участие человека.

Свойство контролепригодности системы.

Успех отладки зависит от того, как спроектирована система, предусмотрены ли свойства, делающие ее удобной для отладки, а также от средств, используемых при отладке. Для проведения отладки проектируемая микропроцессорная система должна обладать свойствами управляемости, наблюдаемости, предсказуемости.

Управляемость - свойство системы, при котором ее поведение поддается управлению, т. е. имеется возможность остановить функционирование системы в определенном состоянии, и затем сновва ее запустить. Наблюдаемость - свойство системы, позволяющее проследить за поведением системы, сменой ее внутренних состояний. Предсказуемость - свойство системы, позволяющее установить систему в состояние, из которого все последующие состояния могут быть предсказаны.

3. Теоретическая часть

Функции средств отладки

Сроки и качество отладки системы зависят от средств отладки. Чем совершеннее приборы, имеющиеся в распоряжении инженера-разработчика, тем скорее можно начать отладку аппаратуры и программ и тем быстрее обнаружить ошибки, локализовать источники, устранение которых обойдется дороже на более позднем этапе проектирования.

Средства отладки должны:

1) управлять поведением системы или/и ее модели на различных уровнях абстрактного представления;

2) собирать информацию о поведении системы или/и ее модели, обрабатывать и представлять на различных уровнях абстракции;
3) преобразовывать системы, придавать им свойства контролепригодности;
4) моделировать поведение внешней среды проектируемой системы.

Под управлением поведением системы или ее модели понимаются определение и подача входных воздействий для запуска или останова системы или ее модели, для перевода в конкретное состояние последних. Чтобы определить место субъективной неисправности, которая может быть внесена на любой стадии проектирования, необходимо уметь собирать информацию о поведении системы и представлять ее в тех формах, которые приняты для данного проекта. Например, это могут быть вре-менные диаграммы, принципиальные электрические схемы, язык регистровых передач, ассемблер и др.

В общем случае нельзя локализовать источник ошибки проектируемой системы, имея информацию о поведении системы только на ее внешних выводах, поэтому проектируемую систему преобразовывают. Например, прежде чем изготовлять однокристальную микроЭВМ с теми или иными "зашивками" ПЗУ, программы отлаживают на эмуляционном кристалле, у которого магистраль выведена на внешние контакты и вместо ПЗУ установлено ОЗУ.

Этапы проектирования микропроцессорных систем

Микропроцессорные системы по своей сложности, требованиям и функциям могут значительно отличаться надежностными параметрами, объемом программных средств, быть однопроцессорными и многопроцессорными, построенными на одном типе микропроцессорного набора или нескольких, и т.д. В связи с этим процесс проектирования может видоизменяться в зависимости от требований, предъявляемых к системам. Например, процесс проектирования МПС, отличающихся одна

от другой содержанием ПЗУ, будет состоять из разработки программ и изготовления ПЗУ.

При проектировании многопроцессорных микропроцессорных систем, содержащих несколько типов микропроцессорных наборов, необходимо решать вопросы организации памяти, взаимодействия с процессорами, организации обмена между устройствами системы и внешней средой, согласования функционирования устройств, имеющих различную скорость работы, и т. д. Ниже приведена примерная последовательность этапов, типичных для создания микропроцессорной системы:
1.Формализация требований к системе.
2.Разработка структуры и архитектуры системы.
3.Разработка и изготовление аппаратных средств и программного обеспечения системы.
4. Комплексная отладка и приемосдаточные испытания.

Этап 1. На этом этапе составляются внешние спецификации, перечисляются функции системы, формализуется техническое задание (ТЗ) на систему, формально излагаются замыслы разработчика в официальной документации.

Этап 2. На данном этапе определяются функции отдельных устройств и программных средств, выбираются микропроцессорные наборы, на базе которых будет реализована система, определяются взаимодействие между аппаратными и программными средствами, временные характеристики отдельных устройств и программ.

Этап 3. После определения функций, реализуемых аппаратурой, и функций, реализуемых программами, схемотехники и программисты одновременно приступают к разработке и изготовлению соответственно опытного образца и программных средств. Разработка и изготовление аппаратуры состоят из разработки структурных и принципиальных схем, изготовления прототипа, автономной отладки.
Разработка программ состоит из разработки алгоритмов; написания текста

исходных программ; трансляции исходных программ в объектные программы; автономной отладки.

Этап 4. см. Комплексная отладка.

На каждом этапе проектирования МПС людьми могут быть внесены неисправности и приняты неверные проектные решения. Кроме того, в аппаратуре могут возникнуть дефекты.

Источники ошибок

Рассмотрим источники ошибок на первых трех этапах проектирования.

Этап 1. На этом этапе источниками ошибок могут быть: логическая несогласованность требований, упущения, неточности алгоритма.

Этап 2. На данном этапе источниками ошибок могут быть: упущения функций, несогласованность протокола взаимодействия аппаратуры и программ, неверный выбор микропроцессорных наборов, неточности алгоритмов, неверная интерпретация технических требований, упущение некоторых информационных потоков.

Этап 3. На этом этапе источниками ошибок могут быть: при разработке аппаратуры - упущения некоторых функций, неверная интерпретация технических требований, недоработка в схемах синхронизации, нарушение правил проектирования; при изготовлении прототипа - неисправности комплектующих изделий, неисправности монтажа и сборки; при разработке программных средств - упущения некоторых функций технического задания, неточности в алгоритмах, неточности кодирования.

Каждый из перечисленных источников ошибки может породить большое число субъективных или физических неисправностей, которые необходимо локализовать и устранить. Обнаружение ошибки и локализация неисправности являются сложной задачей по нескольким причинам: во-первых, из-за большого числа неисправностей; во-вторых,

из-за того, что различные неисправности могут проявляться одинаковым образом. Так как отсутствуют модели субъективных неисправностей, указанная задача не формализована. Имеются определенные успехи в области создания методов и средств обнаружения ошибок и локализации физических неисправностей. Эти методы и средства широко используются для проверки работоспособного состояния и диагностики неисправностей дискретных систем при проектировании, производстве и эксплуатации последних.

Субъективные неисправности отличаются от физических тем, что после обнаружения, локализации и коррекции больше не возникают. Однако, как следует из перечня источников ошибок, субъективные неисправности могут быть внесены на этапе разработки спецификации системы, а это означает, что даже после самых тщательных испытаний системы на соответствие ее внешним спецификациям в системе могут находиться субъективные неисправности.

Процесс проектирования - итерационный процесс. Неисправности, обнаруженные на этапе приемосдаточных испытаний, могут привести к коррекции спецификаций, а следовательно, к началу проектирования всей системы. Обнаруживать неисправности необходимо как можно раньше, для этого надо контролировать корректность проекта на каждом этапе разработки.

4. Практичесмкая часть

Проверка правильности проекта

Основные методы контроля правильности проектирования следующие: верификация - формальные методы доказательства корректности проекта; моделирование; тестирование.

Существует много работ по верификации программного обеспечения, микропрограмм, аппаратуры. Однако эти работы носят теоретический

характер. На практике пока используют моделирование поведения объекта и тестирование.

Для контроля корректности проекта на каждом этапе проектирования необходимо проводить моделирование на различных уровнях абстрактного представления системы и проверку правильности реализации заданной модели путем тестирования. На этапе формализации требований контроль корректности особо необходим, поскольку многие цели проектирования не формализуются или не могут быть формализованы в принципе. Функциональная спецификация может анализироваться коллективом экспертов или моделироваться и проверяться в опытном порядке для выявления, достигаются ли желаемые цели. После утверждения функциональной спецификации начинается разработка функциональных тестовых программ, предназначенных для установления правильности функционирования системы в соответствии с ее функциональной спецификацией. В идеальном случае разрабатываются тесты, целиком основанные на этой спецификации и дающие возможность проверки любой реализации системы, которая объявляется способной выполнять функции, оговоренные в спецификации. Этот способ - полная противоположность другим, где тесты строятся применительно к конкретным реализациям. Независимая от реализации функциональная проверка обычно заманчива лишь в теоретическом плане, но практического значения не имеет из-за высокой степени общности.

Автоматизация утомительной работы по составлению тестовых программ не только сокращает продолжительность периода конструирования/отладки за счет получения тестовых программ на этапе конструирования (поскольку они могут быть сгенерированы сразу после формирования требований к системе), но и позволяет проектировщику изменять спецификации, не заботясь о переписывании всех тестовых программ заново. Однако на практике разработке тестов часто присваивают более низкий приоритет по сравнению с проектом, поэтому тестовые

программы появляются значительно позже его завершения. Но даже если детальные тесты оказываются подготовленными, часто практически нецелесообразно запускать их на имитаторе, так как детальное моделирование требует больших затрат средств на разработку программ и времени на вычисление, в результате большая часть работы по отладке должна откладываться до момента создания прототипа системы.

После обнаружения ошибки должен быть локализован ее источник, чтобы провести коррекцию на соответствующем уровне абстрактного представления системы и в соответствующем месте. Ложное определение источника ошибки или проведение коррекций на другом уровне абстрактного представления системы приводит к тому, что информация о системе на верхних уровнях становится ошибочной и не может быть использована для дальнейшей отладки при производстве и эксплуатации системы. Например, если неисправность внесена в исходный текст программы, написанной на языке ассемблера, а коррекция проведена в объектном коде, то дальнейшая отладка программы ведется в объектном коде; при этом все преимущества написания программы на языке ассемблера сводятся на нет.

Автономная отладка

Процесс отладки прототипа проектируемой системы должен начинаться с отладки аппаратуры и отладки программ.

Отладка аппаратуры предполагает тестирование отдельных устройств микропроцессорной системы - процессора, ОЗУ, контроллеров, блока питания, генератора тактовых импульсов путем подачи тестовых входных воздействий и приема ответных реакций. Тестовые входные воздействия и ответные реакции определяются, исходя из спецификаций на устройства, а также структурных схем устройств. При этом проверяются реальная аппаратура прототипа, спецификации, структурные схемы и отлаживаются тесты. После отладки отдельных устройств проверяется их взаимодействие.

Процессор системы работает с шинами адресов, данных и управления. Анализируя их сигналы, можно проконтролировать выполнение программы в процессоре.

Поскольку ША и ШД синхронные, их работу лучше всего проверить с помощью методов логических состояний. Перед анализом последовательностей данных на этих шинах необходимо удостовериться в том, что сигналы, управляющие взаимодействием процессора с другими устройствами, выдаются в соответствующем порядке. Поскольку ШУ состоит из линий, работающих асинхронно, необходимо просматривать сигналы многих линий в течение одного и того же промежутка времени. Для анализа асинхронной работы линий управления необходимо также наблюдать за сигналами на них при возникновении определенного события, чтобы можно было четко разделить и идентифицировать различные состояния линий. Например, среди сигналов ШУ могут быть сигналы длительностью всего несколько наносекунд, но могут также возникать кратковременные ложные узкие импульсы, вызванные перекрестными помехами или шумами.

После того как доказана работоспособность ШУ, проводится дальнейшая проверка работы аппаратуры при различных режимах адресации процессора и кодах выбираемых данных. Для проверки выполнения процессором инструкций разрабатывается тестовая программа, которая помещается в ОЗУ или ППЗУ. При этом проверяется временная диаграмма сигналов и прохождения данных в системе (как осуществляется передача информации по отношению к строб-сигналам). Если тестовая программа - системный проверяющий тест пройдет успешно, можно утверждать, что автономно аппаратура отлажена.

При автономной отладке аппаратуры могут потребоваться приборы, умеющие: а) выполнять функции аналогового прибора, т. е. измерять напряжение и ток; воспроизводить форму сигнала, подавать импульсы определенной формы и т. д.; б) подавать последовательность сигналов

одновременно на несколько входов в соответствии с заданной временной диаграммой или заданным алгоритмом функционирования аппаратуры, представленным в спецификации на языке высокого уровня, или другим способом; собирать значения сигналов многих линий в течение одного и того же промежутка времени, который определяется задаваемыми, программируемыми событиями - комбинацией или последовательностью сигналов на линиях, например, ложным сигналом на линии; обрабатывать и представлять собранную информацию либо в виде временной диаграммы, либо в виде диаграммы или таблицы логических состояний, либо на языке высокого уровня, например, языке регистровых передач.

Для автономной отладки аппаратуры широко используются осциллографы, вольтметры, амперметры, частотомеры, генераторы импульсов, позволяющие отлаживать аппаратуру на схемном уровне. Чтобы автономно отладить аппаратуру МПС на более высоком уровне, применяют логические анализаторы, генераторы слов, пульты, комплексы диагностирования.

Отладка программ

Отладка программ микропроцессорной системы проводится, как правило, на тех же ЭВМ, на которых велась разработка программ, и на том же языке программирования, на котором написаны отлаживаемые программы, и может быть начата на ЭВМ даже при отсутствии аппаратуры МПС. При этом в системном программном обеспечении ЭВМ должны находиться программы (интерпретаторы или эмуляторы), моделирующие функции отсутствующих аппаратных средств. Так, разработка и автономная отладка программных средств может вестись на больших ЭВМ, миниЭВМ, микроЭВМ, система команд которых не совпадает с системой команд используемого микропроцессора. Кроме того, при отладке программ может отсутствовать внешняя среда микропроцессорной системы, ее также необходимо моделировать.

Проверка корректности программ, т.е. проверка соответствия их внешним спецификациям, осуществляется тестированием. Программы проверяются на функционирование с различными исходными данными. Результаты функционирования программ сравниваются с эталонными значениями.

Отладка программ подразделяется на следующие этапы: планирование отладки; составление тестов и задания на отладку; исполнение программ; информирование о результатах исполнения программ по заданным исходным данным; анализ результатов, обнаружение ошибок и локализация неисправностей.

Существует два способа начального тестирования программ: пошаговый режим и трассировка программ.

В пошаговом режиме программа выполняется по одной команде за один раз, а пользователь анализирует содержимое памяти, регистров и т.д., чтобы проверить, соответствуют ли результаты ожидаемым. Пошаговый режим может быть трудоемким, если средства отладки будут требовать отдельных команд после каждого шага для того, чтобы показать необходимую информацию в понятном для пользователя виде. Имеются средства отладки, автоматически показывающие после каждого шага содержимое регистров процессора и ячеек памяти, используемых в последней команде, и несколько следующих команд. Пошаговый режим является весьма мощным средством предварительного тестирования, так как позволяет обнаруживать неисправности, прежде чем они существенно исказят программу и данные. Кроме того, неоднократно проходя отдельными шагами через один и тот же участок объектной программы, программист может легко изменять содержимое регистров и ячеек памяти, особенно если средства отладки имеют динамически обновляемый дисплей, и тем самым проверить работу программы в разных условиях. Этот интерактивный режим отладки программы позволяет разработчику постоянно упреждать, что будет делать его программа, и оперативно

обнаруживать ошибку. Однако пошаговый режим с автоматическим показом результатов возможен только тогда, когда средства отладки содержат в своем составе дисплей с прямым доступом в память, так как после каждого шага на экране дисплея нужно показывать большой объем информации.

Исполнение программ осуществляется по шагам последовательно во времени и в соответствии с заданиями, содержащимися в операторах. При этом производится переработка значений переменных и определение оператора приемника. Если в ходе исполнения программы регистрируется последовательность операторов, реализуемых на каждом шаге процесса, то получается трасса или маршрут исполнения программы, который для конкретной программы зависит только от значений исходных данных.

Трассировка программ больше пригодна для отладочных средств, имеющих медленный, последовательный терминал. Программа-отладчик выполняет непрерывно команду за командой и выводит содержимое регистров процессора на терминал после каждого шага. Некоторые отладчики выводят также на терминал команды в дизассемблерной форме. Но при этом способе содержимое памяти не выводится на терминал и разработчик должен сам делать выводы об изменениях в ней. Отслеживание программы продолжается автоматически до тех пор, пока не будет остановлено извне. Результатом трассировки программы будут данные на экране дисплея или же в случае использования в качестве терминала печатающего устройства - длинная распечатка с ходом выполнения программы. Программист, анализируя эти данные, может обнаружить ошибки. Трассировка программ не дает, однако, возможности изменять содержимое памяти и регистров и может послужить причиной того, что программа разрушит себя или свои данные прежде, чем отслеживание будет остановлено.

Отдельные участки программы после проверки, используя пошаговый режим или трассировку, можно объединить и проверить с помощью

установки контрольных точек, вводимых в программу и прерывающих ее исполнение, для передачи управления программе-отладчику. По контрольным точкам можно по желанию выполнить избранные участки программы и проанализировать результаты. Контрольные точки устанавливаются обычно для конкретной команды, но в некоторых системах предусматриваются прерывания программы при чтении или записи данных в определенные ячейки памяти. Возможны и более сложные условия прерывания программы.

Расстановка контрольных точек предполагает, что программист связывает с ней точный адрес памяти. Для некоторых отладчиков программист задает абсолютный шестнадцатеричный адрес. Последние отладчики допускают символьные значения адресов, которые программист определяет в исходной программе; это позволяет значительно экономить время, распечатывая после каждого редактирования и транслирования программы новую копию листинга.

При тестировании можно планировать проверку всех возможных маршрутов исполнения программы для разных исходных переменных. Однако это реализуемо только для очень простых программ небольшого объема при малых диапазонах изменения исходных данных. Поэтому при планировании отладки программ применяют критерии полноты тестирования, которые, однако, не гарантируют полной проверки программ. Выбор критерия зависит от наличия ресурсов для тестирования и структурной сложности отлаживаемой программы. Критерии характеризуются глубиной контроля программ и объемом проверок.

В процессе отладки основная часть неисправностей в программах обнаруживается и затем устраняется. Однако всегда возможен пропуск нескольких неисправностей.

Средства отладки программ должны:

а) управлять исполнением программ (останавливать, изменять порядок, запускать и т. д.);

б) собирать информацию о ходе выполнения программы;

в) обеспечивать обмен информацией (диалог) между программистом и ЭВМ на уровне языка программирования;

г) моделировать работу отсутствующих аппаратных средств микропроцессорной системы.

Комплексная отладка микропроцессорных систем

Как правило, микропроцессорная система - это система реального времени, т. е. корректность ее функционирования зависит от времени выполнения отдельных программ и скорости работы аппаратуры. Поэтому система считается отлаженной после того, как рабочие программы правильно функционируют на действительной аппаратуре системы в реальных условиях. Дополнительным свойством, которым должны обладать средства комплексной отладки по сравнению со средствами автономной отладки, является возможность управления поведением МПС и сбора информации о ее поведении в реальном времени.

Простой контроллер для синхронной передачи данных в ВУ по последовательной линии связи (последовательный интерфейс) представлен на рис. 3.7.

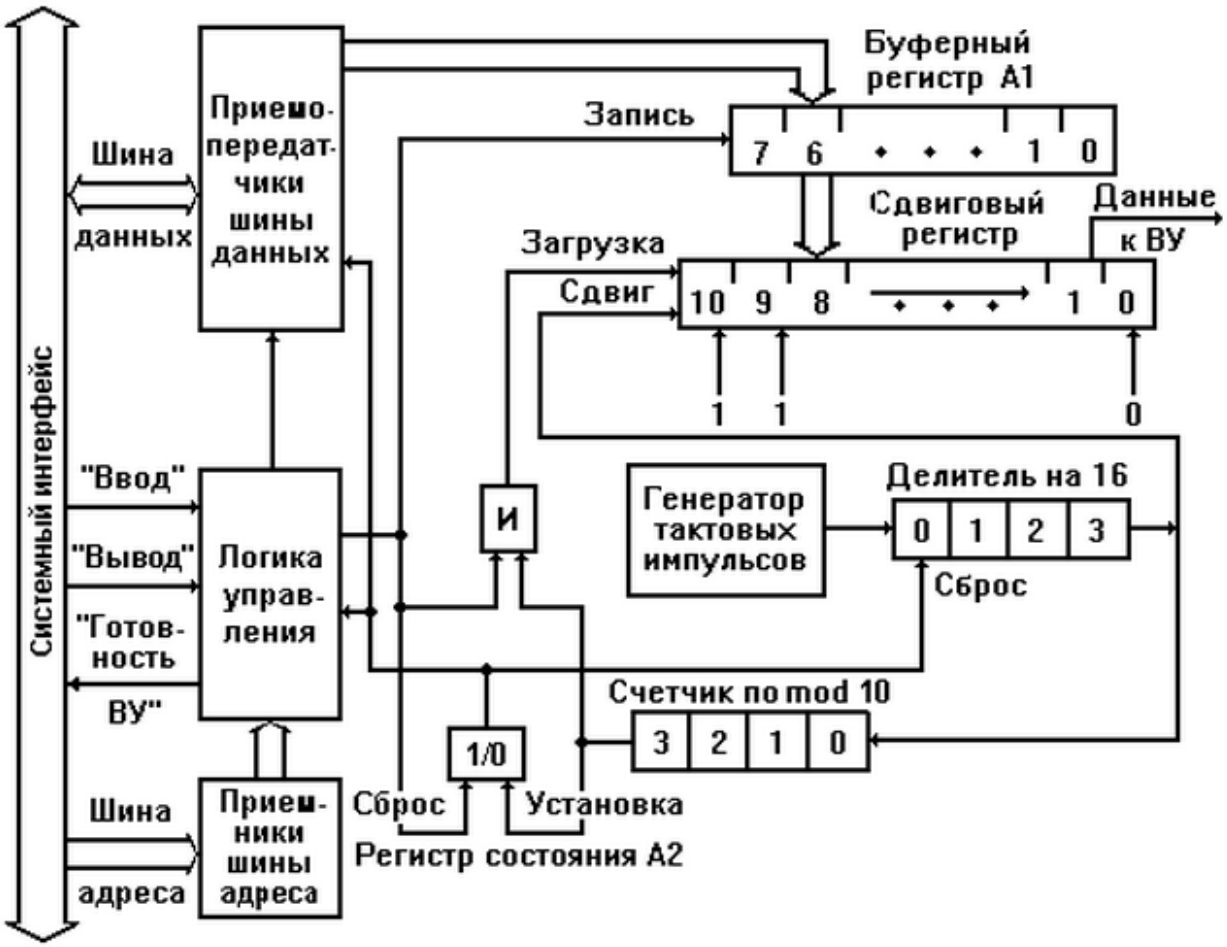

Рис. 3.7. Контроллер последовательной синхронной передачи

Восьмиразрядный адресуемый буферный регистр контроллера А1 служит для временного хранения байта данных до его загрузки в сдвиговый регистр. Запись байта данных в буферный регистр с шины данных системного интерфейса производится так же, как и в параллельном интерфейсе (см. Параллельная передача данных и рис. 3.5), только при наличии единицы в одноразрядном адресуемом регистре состояния контроллера А2. Единица в регистре состояния указывает на готовность контроллера принять очередной байт в буферный регистр. Содержимое регистра А2 передается в процессор по одной из линий шины данных системного интерфейса и используется для формирования управляющего сигнала системного интерфейса "Готовность ВУ". При записи очередного байта в буферный регистр А1 обнуляется регистр состояния А2.

Программа записи байта данных в буферный регистр аналогична программе из примера 2.1 за исключением команды перехода: вместо команды JNZ m1 (переход, если не ноль) необходимо использовать команду JZ m1 (переход, если ноль).

Преобразование данных из параллельного формата, в котором они поступили в буферный регистр контроллера из системного интерфейса, в последовательный и передача их на линию связи производятся в сдвиговом регистре с помощью генератора тактовых импульсов и двоичного трехразрядного счетчика импульсов следующим образом.

Последовательная линия связи контроллера с ВУ подключается к выходу младшего разряда сдвигового регистра. По очередному тактовому импульсу содержимое сдвигового регистра сдвигается на один разряд вправо и в линию связи "Данные" выдается значение очередного разряда. Одновременно со сдвигом в ВУ передается по отдельной линии "Синхронизация" тактовый импульс. Таким образом, каждый передаваемый по линии "Данные" бит информации сопровождается синхронизирующим сигналом по линии "Синхронизация", что обеспечивает его однозначное восприятие на приемном конце последовательной линии связи.

Количество переданных в линию тактовых сигналов, а следовательно, и переданных бит информации подсчитывается счетчиком тактовых импульсов. Как только содержимое счетчика становится равным 7, т. е. в линию переданы 8 бит (1 байт) информации, формируется управляющий сигнал "Загрузка", обеспечивающий запись в сдвиговый регистр очередного байта из буферного регистра. Этим же управляющим сигналом устанавливается в "1" регистр состояния. Очередным тактовым импульсом счетчик будет сброшен в "0", и начнется очередной цикл выдачи восьми битов информации из сдвигового регистра в линию связи.

Синхронная последовательная передача отдельных битов данных на линию связи должна производиться без какого-либо перерыва, и

следующий байт данных должен быть загружен в буферный регистр из системного интерфейса за время, не превышающее времени передачи восьми битов в последовательную линию связи.

При записи байта данных в буферный регистр обнуляется регистр состояния контроллера. Нуль в этом регистре указывает, что в линию связи передается байт данных из сдвигового регистра, а следующий передаваемый байт данных загружен в сдвиговый регистр.

Контроллер для последовательного синхронного приема данных из ВУ состоит из тех же компонентов, что и контроллер для синхронной последовательной передачи, за исключением генератора тактовых импульсов.

Тенденция развития средств отладки микропроцессорных систем состоит в объединении свойств нескольких приборов в одном комплексе, в создании универсальных средств, пригодных для автономной отладки аппаратуры, генерации и автономной отладки программ и комплексной отладки системы. Эти средства позволяют вести разработку и отладку, постепенно усложняя аппаратуру и программы. При этом разработка, изготовление и отладка планируются поэтапно с нарастанием сложности; новая, неотлаженная аппаратура и программа вводятся в создаваемую систему, присоединяются к проверенной ее части.

Если отладка программ ведется с использованием эмуляционного ОЗУ, а затем изготовляются микросхемы ПЗУ, то микропроцессорная система должна быть протестирована.

Средства отладки на последних этапах не должны влиять на правильность функционирования системы, вносить задержки, дополнительные нагрузки.

При комплексной отладке наряду с детерминированным используется статистическое тестирование, при котором МПС проверяется при изменении исходных переменных в соответствии со статистическими законами работы источников информации. Полнота контроля

работоспособности проектируемой системы возрастает за счет расширения диапазона возможных сочетаний переменных и соответствующих им логических маршрутов обработки информации.

Существуют пять основных приемов комплексной отладки микропроцессорной системы:

1) останов функционирования системы при возникновении определенного события;

2) чтение (изменение) содержимого памяти или регистров системы;

3) пошаговое отслеживание поведения системы;

4) отслеживание поведения системы в реальном времени;

5) временное согласование программ.

Комплексная отладка завершается приемосдаточными испытаниями, показывающими соответствие спроектированной системы техническому заданию. Для проведения комплексной отладки МПС используют логические анализаторы и комплексы: оценочные, отладочные, развития микропроцессоров, диагностирования, средств отладки.

5. Заключение

Заключение

Отметим, что в представленном курсовом проекте даны основные, базовые понятия микропроцессорной техники: определение, классификация, логическая структура, система и формат команд, подсистема памяти и теоретические аспекты проектирования МПС.

Будущее микропроцессорной техники связано сегодня с двумя новыми направлениями - нанотехнологиями и квантовыми вычислительными системами. Эти пока еще главным образом теоретические исследования касаются использования в качестве компонентов логических схем молекул и даже субатомных частиц: основой для вычислений должны служить не электрические цепи, как сейчас, а положение отдельных атомов или направление вращения электронов. Если "микроскопические" компьютеры

будут созданы, то они обойдут современные машины по многим параметрам.

6. Список испльзованной литературы

1. Велихов Е. П. Сергей Алексеевич Лебедев // Информационные технологии и вычислительные системы. — 2002. — № 3. — С. 31–35.

2. Бурцев В. С. Параллелизм вычислительных процессов и развитие архитектуры ЭВМ. — М.: Торус пресс, 2006.

3. Борисов Ю. И. Проектные центры компьютеростроения в программах обеспечения национальной безопасности // Матер. конф. «Перспективы развития высокопроизводительных архитектур. История, современность и будущее отечественного компьютеростроения». — 2008. — № 1. — С. 8–14.

4. Бабаян Б. А. История развития архитектур вычислительных машин //

Ершовские лекции по информатике. — Новосибирск: Изд-во Ин-та информатики им. А. П. Ершова СО РАН, 2009.

5. Ким А. К., Перекатов В. И., Сахин Ю. Х. Развитие и реализация архитектуры вычислительных комплексов серии «Эльбрус» для решения задач ракетно-космической обороны // Вопросы радиоэлектроники. Сер. ЭВТ. — 2010. — Вып. 3. — С. 5–17.

6. Фельдман В. М. Многопроцессорные вычислительные комплексы отечественной разработки // Сборник научных трудов ИМВС РАН «Высокопроизводительные вычислительные системы и микропроцессоры». — 2003. — С. 3–15.

7. Ким А. К., Волконский В. Ю., Груздов Ф. А. и др. Микропроцессорные вы-

числительные комплексы с архитектурой «Эльбрус» и их программное обеспечение и др. // Вопросы радиоэлектроники. Сер. ЭВТ. — 2009. —

Вып. 3. — С. 5–37.

8. Исаев М. В., Кожин А. С., Костенко В. О. и др. Двухъядерная гетероген-

ная система на кристалле «Эль б рус-2С+» // Вопросы радиоэлектроники. Сер. ЭВТ. — 2012. — Вып. 3. — С. 42–52.

9. Ким А. К. Российские универсальные микропроцессоры и вычислительные комплексы высокой производительности: результаты и взгляд в будущее // Вопросы радиоэлектроники. Сер. ЭВТ. — 2012. — Вып. 3. — С. 5–13.

1 0. Ким А. К., Фельдман В. М. Вопросы создания суперЭВМ на основе современной платформы «Эльбрус» // Приборы. — 2009. — № 1. — С. 36–46.

1 1. Волконский В. Ю., Ким А. К., Перекатов В. И., Фельдман В. М. Микропроцессоры и вычислительные комплексы компании МЦСТ // Электроника. — 2008. — № 8. — С. 62–70.

1 2. Мелехин В. Ф., Павловский Е. К. Вычислительные машины, системы и сети. — М.: ACADEMIA, 2010.

1 3. The SPARC architecture manual: Version 8. — Englewood Cliffs, N.J.: Prentice Hall, 1992.