


Шубинский И.Б., Шебе Х.

СИСТЕМАТИЧЕСКИЙ ПОДХОД К ЗАЩИТЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОТ СБОЕВ АППАРАТУРЫ

В статье описывается суть сбойных ошибок, их возникновение и воздействие на результаты функционирования системы. Рассматриваются методы обнаружения сбойных ошибок. Описываются методы защиты от сбоев и предлагается комплексный подход к обеспечению устойчивости системы к сбоям. Принцип, который излагается в статье, показывает каким основным свойством должна обладать устойчивая к сбоям система. Далее приводится пример и заключение по результатам исследования.

Ключевые слова: *сбои, сбойные ошибки, помехи, искажение данных, искажение последовательности программы, устойчивость к сбоям.*

1. Введение

Обычно в теории надежности и безопасности много внимания уделяют отказам, в особенности опасным. Однако помимо них существуют другие события, которые на первый взгляд кажутся гораздо менее важными. Это – сбои [1,2].

Сбои определяются следующим образом. Сбой функционального характера – событие, заключающееся в однократном искажении перерабатываемой или хранящейся в информационной технике информации, возникающее под воздействием внутренних или внешних дестабилизирующих факторов (помех) [1].

Тема становится более актуальной по двум причинам. Во-первых, миниатюризация вычислительной техники приводит к растущей чувствительности аппаратных средств. Во-вторых, ослабление магнитного поля земли приводит к увеличению попадания на землю заряженных космических частиц. В результате частота сбоев на три и более порядка превышает частоту отказов технических средств [1]. В данной статье предлагается подход к созданию устойчивой по отношению к сбоям системы.

2. Сбои

В информационных системах (ИС) сбои, которые относятся к категории перемежающихся отказов, не оказывают существенного влияния на надежность функционирования ИС. Это объясняется следующими основными причинами:

Многие ИС относятся к категории критически важных систем. Поэтому при создании средств информационной техники применяются специальные меры по стабилизации в приемлемых границах температуры и влажности, обеспечивается качественное кондиционирование помещений и стативов, тщательное проектирование электрических режимов, исключение событий «гонок и состязаний сигналов» и др.;

Многие информационные системы работают в реальном масштабе времени и в директивные сроки выполняют строго определенные информационные функции. Для этих систем приоритетное влияние на результаты функционирования оказывают сбои функционального характера, т.е. такие сбои техники, которые при определенных обстоятельствах могут привести к ошибкам в управлении и иметь серьезные последствия для работы всей системы в целом и даже для окружающей среды;

Сбои характеризуются тем, что они существуют очень короткое время. Они возникают в аппаратных средствах и исправляются сами по себе, т.е. аппаратные средства возвращаются в исходное физическое состояние, но не обязательно в исходное функциональное состояние.

Сбои аппаратных средств могут быть вызваны следующими причинами:
Помехи по входам и цепи питания (рис.1.)

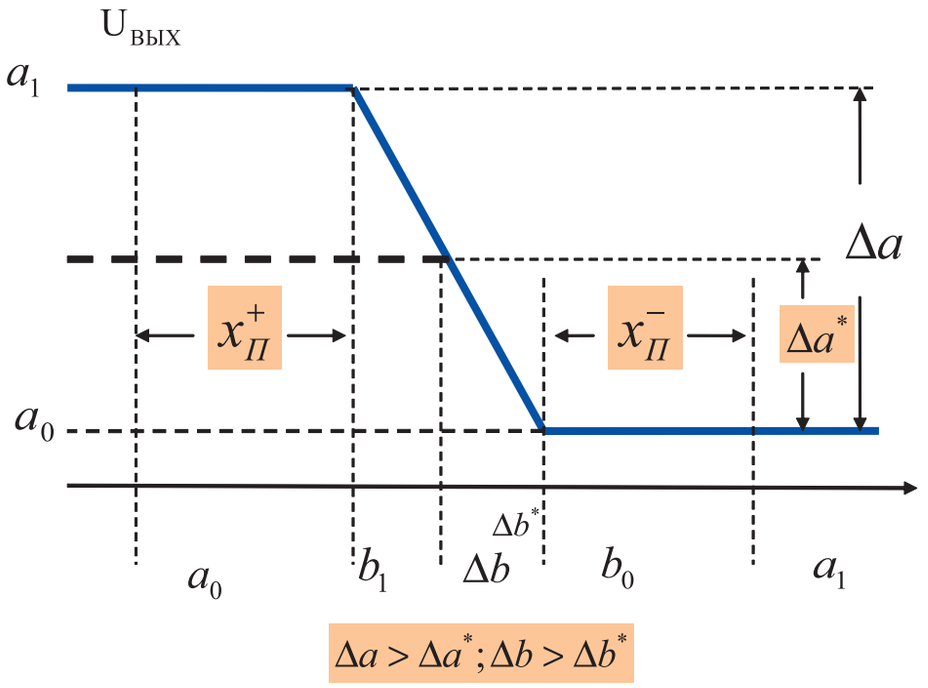

Рис. 1. Типовая передаточная характеристика цифровой интегральной схемы

Реальная передаточная характеристика схемы может быть аппроксимирована кусочно-линейной функцией. На рис. 1 приведен пример типовой передаточной характеристики и указаны ее параметры: a_i (i =1,0) – выходной уровень напряжения логической единицы или нуля; b_i – пороговое значение входного сигнала при переключении типа 1→ 0 (i =1) или типа 0 → I (i = 0); $\bar{b} = \frac{b_1 + b_0}{2}$; возможная ширина зоны переключения $\Delta b = /b_1 - b_0/$. На типовой передаточной характеристике выделяются три области, соответствующие различным состояниям схемы. Область I – закрытое состояние, область III – открытое состояние схемы, а промежуточная область II – состояние пере-

ключения. Под воздействием отпирающей помехи возможен переход схемы из области I в область III, т.е. неправильное формирование на выходе схемы кода „I“ вместо кода „0“ (сбой типа I → 0). Под воздействием запирающей помехи возможен сбой типа 0 → I.

Из рис. 1 следует, что при заданной величине логического перепада Δa технологический разброс пороговых напряжений b_1 и b_0 приводит к уменьшению максимально допустимых амплитуд отпирающей x_{Π}^{+} и запирающей x_{Π}^{-} помех, а, следовательно, к снижению статической помехоустойчивости схемы. Аналогичный результат имеет место при уменьшении логического перепада Δa, который осуществляется в интересах повышения быстродействия схем. Чем меньше величина логического перепада $\Delta a^* < \Delta a$, тем меньше время переключения схемы $\Delta b^* < \Delta b$ и, следовательно, тем выше ее быстродействие.

Экспериментальные данные по сбоям интегральных схем отсутствуют. Для оценки характеристик сбоев применяются экспериментально-расчетные методы прогнозирования, основанные на экспериментальных данных о входных сигналах и помехах, передаточных и амплитудно-частотных характеристиках схем. Эти данные являются исходными для расчета диапазона возможных вероятностей сбоя цифрового логического элемента в интегральном исполнении, который часто именуют вентилем по аналогии с транзистором в элементной базе второго поколения. По полученным расчетным значениям сбоев вентилей, известной структуре и реализуемым логическим функциям рассчитываются вероятности и интенсивности сбоев ЦИС.

Помехи электромагнитного характера

Если помехи могут воздействовать на систему через входы, питание и массу, то система также может быть подвержена электромагнитному излучению. Вследствие этого может индуцироваться ток, представляющий собой помеху. Эта помеха действует таким же методом на контакты ИС, как помеха, прямо проходящая по контактам и проводам и как это описано выше.

Пролет радиоактивных частиц

Третий механизм – это воздействие радиоактивных частиц. Обычно на первом шаге гамма-излучение проходит близко к ИС. Это, скорее всего, излучение, возникающее вследствие радиоактивного распада. Гамма-излучение имеет свойство хорошей проходимости через материал. Защищаться от гамма-излучения трудно. Квант гамма-излучения может выбивать альфа частицу, которая как заряженная частица может воздействовать на входы или прямо внутри ИС на регистры и память. Вследствие такого воздействия может исказиться хотя бы бит информации.

Последствия сбоев могут заключаться в следующем:

• *Искажение записанных данных*

В этом случае система работает на основе неправильных данных. В этом случае даже при корректно выполненных функциях возникает ложный результат. Причем этот процесс продолжается в дальнейшем, если ошибка в данных не скорректирована. Ошибка может оказаться в выходных результатах работы ИС, сформировать ложную команду и привести в итоге к опасному отказу.

• *Изменение последовательности программы*

Это значит, что система функцию не выполняет вообще, либо выполняет другую функцию. Возможны также события преждевременного выполнения функции, или недопустимо позднего или неуместного выполнения функции, что в итоге может привести к опасному отказу системы.

3. Обнаружение сбойных ошибок и защита от их опасных последствий

Существует ряд хорошо известных методов защиты от последствий сбоев [1]. В первую очередь требуется наличие методов обнаружения сбойных ошибок. Известны следующие методы обнаружения:

• *Многоканальность.* При этом подходе сравниваются результаты двух или более каналов. Если они различаются, то в устройстве обнаруживается сбой или отказ.

• *Проверочные суммы.* Помимо самих данных запоминается контрольная сумма. Изменения данных приведут к тому, что принятая и вновь сформированная по принятым данным контрольные суммы не совпадают. Этим методом можно обнаружить многие сбои, но далеко не все.

• *Алгоритмы.* Используются алгоритмы, которые толерантны по отношению к сбоям. Однако для ряда задач не удается найти или построить подобные алгоритмы.

• *Многоверсионное программирование.* Создаются две или более версии программного обеспечения, основанные на том же алгоритме. Сравниваются результаты. Так как действие сбоя предполагается различным на обе версии программного обеспечения, сбои должны обнаруживаться.

Эти обычные классические методы защиты от сбоев пригодны для широкого класса ошибок и отказов.

После обнаружения сбоев необходимо с помощью программно-аппаратных средств осуществить их устранение в системе. Однако дополнительные средства как для обнаружения, так и для устранения сбойных ошибок также подвергаются воздействию сбоев. Следовательно, они в свою очередь должны защищаться от сбоев.

4. Систематический подход

Построим сейчас систематический метод защиты от сбоев.

Отказ вследствие сбойной ошибки появляется как путем нарушения логических условий работы системы, так и в виде нарушения точности или пропускной способности работы системы. Эти искаженные промежуточные или выходные результаты запоминаются системой прямым или косвенным образом. Причем косвенное запоминание более опасно, так как оно трудно обнаруживается.

Пример прямого запоминания – это искажение данных, например, параметров системы. Косвенное запоминание появляется, если система выполняет неправильную операцию, что приведет к искаженным данным, которые потом заносятся в память.

Программное обеспечение (ПО) можно представить в форме конечного автомата. Состояние автомата определяется данными, записанными в памяти, которые либо поступили извне (данные датчиков, состояния внешних устройств), либо сформировались при выполнении процессов самого ПО (результаты расчетов, команды от ПО, направленные внешним устройствам и др.). Причем вся информация дискретна и зависит только от всей истории системы, но не от ее будущего. Теперь вычислительную систему можно описать следующим образом. Состояние в определенный момент описывается вектором в пространстве состояний системы, который содержит всю необходимую информацию для однозначного описания системы: все полученные данные, код ПО, значения регистров и т.д. Вектор состояния, хотя и очень большого размера, но конечен. Тогда ПО и цифровую систему можно описать с помощью модели Марковской цепи. Это объясняется тем, что так же как и в вычислительной системе, в модели Марковской цепи поведение системы в будущем зависит от настоящего и не зависит от того, когда и каким образом система перешла в это состояние.

Задача защиты от сбойных ошибок сводится, таким образом, к двум подзадачам:
1. Защита зафиксированного состояния от искажения сбоями;
2. Защита перехода в следующее состояние от искажения сбоями.

Для решения этих подзадач можно использовать, применительно к каждой из них, следующие меры:

1. *Проверочные суммы, избыточная запись.* С помощью таких методов можно детектировать искажение данных о состоянии и, если метод запоминания содержит избыточную информацию, можно даже восстановить искаженную информацию. Это возможно, если используется проверочный код с восстановлением, как, например, для CD, или информация запоминается дважды с проверочной суммой для каждой версии отдельно. Для того чтобы система работала эффективно, можно встроить машину состояний, которая отражает все важные аспекты процесса, в особенности различие между опасными и безопасными состояниями. Важно найти или сформировать машину состояний минимального объема. В стандарте EN 50128 (2011) упоминается в приложении D.24 метод Finite State Machines [3]. Это мероприятие как раз мотивировано защитой от сбойных ошибок.

2. *Многоверсионное программирование.* Если программное обеспечение спрограммировано в различных версиях именно так что они проводят расчеты различными методами, то сбои не смогут исказить обработку данных в обеих версиях так, чтобы результаты совпали. Также возможно использовать тот же код на различных компьютерах, на которых он транслируется разным образом в разных исполняемых ПО. Если использовать машины состояний, то перед каждым переходом состояний можно установить проверку, которая использует различный алгоритм.

5. Пример

В этом разделе приводится пример системы, в которой применяется систематический подход защиты от сбоев, представленный выше. Предполагается, что для проведения расчетов требуется безопасный компьютер. Причем расчет должен проводиться циклически. Например, каждые 100 миллисекунд вычисляется тормозной путь, с целью проверки того, что поезд будет останавливаться перед светофором с запрещающим показанием.

На первом шаге следует определить тот запас данных, который ясно описывает состояние системы. В примере это позиция, скорость, замедление и кривая, описывающая способность торможения поезда. Далее важно время или номер цикла. Очевидно, эти данные следует записать в память вместе с проверочными суммами. Тогда можно обнаружить искажения данных, вызванные сбоями. Если важно не только обнаружение, но и восстановление корректных данных и тем самым восстановление корректного состояния, то следует использовать либо избыточную запись данных, либо использование корректирующих кодов. В первом случае данные запоминаются дважды со своим проверочным кодом. Тогда легко установить, какая версия данных искажена. Во втором случае следует уделить внимание тому обстоятельству, что при большом количестве сбоев станет вероятным ложное восстановление, т.е. восстановление данных в неправильном состоянии. Во всяком случае требуется доказать расчетом, что мероприятия защиты от сбоев обнаруживают ошибки с достаточно высокой вероятностью, так что вероятность незамеченных искажений мала.

На втором шаге следует защищаться от последствий сбоев при переходе из одного состояния в другое. Самый простой вариант – это определение следующего состояния несколько раз. Расчет, проведенный дважды, позволяет обнаруживать ошибку. Если в таком случае повторить расчет третий раз, можно корректировать ошибку путем перехода в то состояние, которое определилось большинством расчетов.

При этом повторные расчеты проводятся на том же процессоре и той же самой версией программного обеспечения, как первый. Это возможно потому, что сбои не приведут к постоянным отказам. Надо отметить, что описанные меры не помогают при постоянных или систематических отказах.

В связи с переходами следует рассматривать программное обеспечение, которое проводит расчеты для определения последующего состояния. Если сбойная ошибка искажает программное обеспечение, то оно и остается в искаженном виде в памяти, хотя сама память при следующей записи программного обеспечения ее бы запомнила правильно. Поэтому программное обеспечение следует записать в память вместе с проверочной суммой. Более того, при каждом применении, т.е. и при повторном расчете перехода системы в следующее состояние следует проверить целостность программного обеспечения (или его части) с помощью проверочной суммы.

6. Заключение

В этой статье мы исследовали сбойные отказы. Мы описали причины сбойных отказов и методы защиты от них. Отсюда мы вывели систематический подход, который состоит в двух шагах. На первом шаге система представляется как Марковский автомат, состояние которого запоминается защищенным методом, например с помощью проверочных сумм. На втором шаге строится надежный метод перехода между состояниями, например, с помощью избыточных расчетов.

Подход проиллюстрирован на примере.

Литература

1. **Шубинский И.Б.** Функциональная надежность информационных систем, Москва, 2012.

2. **Haiyu Qi, Sanka Ganesan, Michael Pecht,** No-fault-found and intermittent failures in electronic products, Microelectronics Reliability 48 (2008) 663–674.

3. EN 50128, Railway applications — Communication, signaling and processing systems — Software for railway control and protection systems, 2011.