


Харлап С.Н., Сивко Б.В.

РАЗРАБОТКА ВЫСОКОНАДЕЖНЫХ СИСТЕМ НА ОСНОВЕ МЕТОДА ВЗАИМНОЙ ПРОВЕРКИ АКСИОМАТИЧЕСКИХ БАЗИСОВ

На основе метода взаимной проверки аксиоматических базисов разработаны безопасные микропроцессорные системы, устойчивые к отказам. Проведены их испытания на безопасность и отказоустойчивость с помощью комплекса имитационных испытаний на безопасность (КИИБ). Выявлено, что усиление проверяющего аксиоматического базиса позволяет улучшить показатели отказоустойчивости. Показано, что взаимная проверка аксиоматических базисов позволяет разрабатывать отказоустойчивые и безопасные системы с целевыми свойствами.

Ключевые слова: *функциональная безопасность, формальные методы, отказоустойчивость, обнаружение отказов, критически важные объекты информатизации.*

Введение

К современным системам, критичным к безопасности, предъявляются повышенные требования как по надежности, так и по безопасности функционирования. Для обеспечения соответствующего качества требуется проведение мероприятий, позволяющих улучшить показатели отказоустойчивости, одной из задач которых является обнаружение отказов с последующей реакцией, в виде перехода в безопасное состояние или запуска процесса восстановления. Так как в настоящее время данная проблема не имеет единого решения, то актуальна разработка методов и средств, позволяющих эффективно решать задачу обнаружения отказов [1].

В статье рассматриваются особенности разработки и испытаний на безопасность функционирования аппаратно-программного комплекса (АПК), построенного на основе взаимной проверки аксиоматических базисов [2].

Для оценки эффективности метода результаты его применения должны быть проверены на практике. В настоящее время для этой цели применяется имитационное моделирование как способ, позволяющий в лабораторных условиях провести с минимальными затратами полный цикл разработки и верификации. В связи с этим в качестве инструмента моделирования выбран программный комплекс имитационных испытаний на безопасность (КИИБ), который предназначен для проведения имитационных испытаний на функциональную безопасность в соответствии с IEC 61508, EN 50126, ОСТ 32.146 микропроцессорных систем управления ответственными технологическими процессами [3]. С помощью КИИБ возможно внесение различного вида отказов технических средств, а также анализ последующего поведения рассматриваемой системы.

Объектом исследования является АПК, в состав которого входит микроконтроллер PIC16F877A.

При разработке отказоустойчивых систем рекомендуемой практикой является защита от определенных видов отказов, которые являются известными и характерными для рассматриваемого аппаратного обеспечения. В данном случае выбор множества отказов, от которых требуется защита, обусловлен требованиями стандарта IEC 61508-2:2010. В связи с этим рассматриваются отказы используемых ячеек памяти: отказы постоянных единицы и нуля (stuck-at faults), а также отказы короткого замыкания (bridging faults).

Исходя из требований метода, изначально требуется наличие двух базисов, которые являются независимыми и диверситетными [4]. В качестве первого базиса *A* выбрана корректная работа в условиях отказов одного байта оперативной памяти (*X*). Второй базис *B* представляет собой аналогичные условия для аккумулятора микроконтроллера (*W*) и флага переноса (*C*). Взаимная проверка показана на рисунке 1, где базис *A* проверяет базис *B* функцией s_1, а *B* проверяет *A* функцией s_2. В системе, выполняющей заданную задачу, присутствует целевая функция *f*, которая выполняется на обоих базисах.

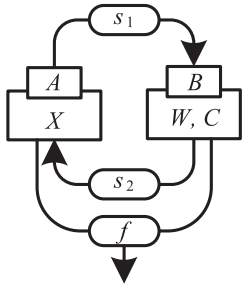

Рис.1. Взаимная проверка базисов

Разработка безопасной системы

В качестве сигнала обнаружения отказа как результат проверки базиса удобно выбрать две цифровые линии (назовем их *SAFE_A* и *SAFE_B*), которые могут находиться в состояниях логического 0 или 1 и относиться к портам микроконтроллера. В случае успешной работы системы на *SAFE_A* и *SAFE_B* передается переменный сигнал (0 и 1). Когда один из базисов нарушается, то соответствующий выход принимает постоянное значение 0.

Внешний сигнал описанного типа прост в реализации и обработке, а также на его основании можно использовать внешние аппаратные средства, позволяющие разрешить ситуацию в случае отказа, например, перевести систему в безопасное состояние.

Алгоритм построен следующим образом:

1. Установить *SAFE_A*=0.
2. Проверка с помощью базиса *A* базиса *B*.
3. Если выполняется *B*, то установить *SAFE_A*=1.
4. Установить *SAFE_B*=0.
5. Проверка с помощью базиса *B* базиса *A*.
6. Если выполняется *A*, то установить *SAFE_B*=1.
7. Переход к пункту 1.

Таким образом, рассматриваемая система выполняет только постоянную проверку базисов и в зависимости от результатов формирует выходные сигналы *SAFE_A* и *SAFE_B*. Каждый из сигналов может быть либо переменным (~), либо постоянным (0), по которым диагностируется наличие отказов (таблица 1).

Таблица 1 **– Состояния при диагностике отказов**

№	*SAFE_A*	*SAFE_B*	Описание
1	~	~	**Отказы отсутствуют, оба базиса выполняются**
2	~	**0**	**Базис *B* выполняется, базис *A* нарушен**
3	**0**	~	**Базис *A* выполняется, базис *B* нарушен**
4	**0**	**0**	**Один базис нарушен или оба**

Для выполнения условий безопасности в случае нарушения базиса *A* система должна гарантированно переходить в состояние 2 или 4 согласно таблице 1. Аналогично, при нарушении базиса *B* должен быть переход в состояние 3 или 4.

Система считается отказоустойчивой для состояний 1, 2 и 3 таблицы 1. В этом случае один из базисов выполняется, а реализованная на соответствующем базисе функциональность рассматриваться как исправная.

Подпрограмма на языке ассемблер микроконтроллера PIC16F877A [5], выполняющая проверку базиса *A* на основании базиса *B*, представлена в таблице 2. Данная подпрограмма является реализацией пунктов 5 и 6 алгоритма. Так как выполнение подпрограммы занимает конечное время, которое гарантированно выше определенного предела, то на выходе *SAFE_B* в случае корректности базисов будет получен переменный сигнал.

Таблица 2 **– Программная проверка базиса *A***

Действие	Программа
Проверка отказа постоянной единицы (*stuck-at-1 fault*)	clrf X movf X,0 addlw 0x0ff btfsc STATUS,C return
Проверка отказа постоянного нуля (*stuck-at-0 fault*)	movlw 0x0ff movwf X movf X,0 addlw 0x01 btfss STATUS,C return
Проверка отказа монтажного И/ИЛИ для соседних бит (*bridging fault*)	movlw 0x0aa movwf X movf X,0 addlw 0x055 addlw 0x01 btfss STATUS,C return
Сигнализация успешности проверки базиса	bsf SAFE_B return

***Таблица 3* – Результаты испытаний безопасной системы**

Отказы	Состояния выходных сигналов			
	Испытания 1		Испытания 2	
	SAFE_A	SAFE_B	SAFE_A	SAFE_B
–	~	~	~	~
$SA1(W_{1,3})$	0	0	0	~
$SA1(W_{0,2,4..7})$	0	0	0	0
$SA0(W)$	0	0	0	0
$B(W)$	0	0	0	0
$SA0(C)$ $SA1(C)$	0	0	0	0
$B(C)$	0	~	0	~
$SA0(X)$	0	0	0	0
$SA1(X)$	0	0	0	0
$B(X)$	0	0	0	0

Примечание: 0 – переход в постоянный сигнал 0; ~ – переменный сигнал согласно требованиям; W_i – i-й бит регистра W.

Для проверки способности разработанной программы к обнаружению отказов рассматриваемый АПК был реализован в КИИБ, где проводилось имитационные испытания. Согласно базисам и обнаруживаемым отказам был составлен список отказов, подлежащих проверке:

– $SA0(W)$, $SA1(W)$, постоянные 0 и 1 аккумулятора, 16 отказов;

– $B(W)$, монтажное И/ИЛИ аккумулятора (14 отказов);

– $SA0(C)$, $SA1(C)$, постоянные 0 и 1 для флага C (2 отказа);

– $B(C)$, монтажное И/ИЛИ регистра *STATUS*, затрагивающего флаг C (2 отказа);

– $SA0(X)$, $SA1(X)$, постоянные 0 и 1 ячейки памяти X (16 отказов);

– $B(X)$, монтажное И/ИЛИ ячейки памяти X (14 отказов).

В программу испытаний было включено внесение каждого единичного отказа, а также проверка работы без отказов. По полученным графикам сигналов *SAFE_A* и *SAFE_B* определялась успешность реакции системы на отказ. Результаты имитационного моделирования по приведенному списку отказов представлены в таблице 3 (испытания 1). Испытания 2 таблицы 3 проводились с измененным для повышения отказоустойчивости программным обеспечением (ПО) (об этом речь пойдет позже).

Испытания показали, что при внесении большинства из описанных отказов оба сигнала *SAFE_A* и *SAFE_B* переходили в постоянное значение 0. Прежде всего, важным результатом является то, что проверка соответствующего базиса всегда проходила успешно, т.е. если некий базис проверяет второй базис, а последний не выполняется из-за наличия отказа, то первый всегда сигнализирует о проблеме. Другими словами, изначально было математически доказано, что в случае внесения одного из рассматриваемых отказов система всегда перейдет в состояние, сигнализирующее о проблеме соответствующего базиса. Это подтвердилось на практике имитационного моделирования, когда при нарушении базиса A сигнал *SAFE_B* всегда переходил в состояние 0 (также при нарушении B сигнал *SAFE_A* переходил в 0). Тем самым было показано, что в случае отказов система всегда переходит в безопасное состояние, что говорит об успешности применения метода с точки зрения создания безопасной системы.

Вместе с тем, в большинстве случаев наблюдалось, что при нарушении базиса A сигнал *SAFE_A* также переходил в состояние 0. Данное поведение объясняется тем, что проверка базиса B нарушенным базисом A оказалась настолько чувствительной, что она просигнализировала о нарушении базиса B несмотря на то, что последний оставался в корректном состоянии. Исключением являлись отказы $B(C)$, которые не повлияли на выходной сигнал, так как данный ресурс (соседний бит к флагу C регистра *STATUS*) в логике проверки не задействован.

Дополнительным выводом из анализа результатов является повышение отказоустойчивости, что имеет место тогда, когда один из сигналов *SAFE_A/SAFE_B* находится в постоянном (0), а второй является (~). Результаты испытаний показали, что в большинстве случаев отказоустойчивость не повысилась, что объясняется как чувствительностью проверки базисов, так и тем фактом, что для повышения отказоустойчивости не было проведено дополнительных мероприятий.

Повышение отказоустойчивости усилением базиса

Для улучшения показателей отказоустойчивости можно построить ПО таким образом, чтобы система была менее подвержена отказам, т.е. базис должен быть усиленным. Это возможно тогда, когда ПО использует меньшее число ресурсов. Для этой цели был изменен код программы проверки базиса A на отказы постоянной единицы (таблица 4).

С модифицированным ПО была проведена вторая серия испытаний, результаты которых показаны в таблице 3 (испытания 2). ПО было изменено таким образом, что в качестве ресурса использовался только старший полубайт аккумулятора, а младший не был задействован. Как следствие, для двух отказов $SA1(W_1)$ и $SA1(W_3)$ система начала диагностировать проблему только у одного из базисов, что говорит об улучшении показателей отказоустойчивости.

Таблица 4 – Измененная проверка базиса A на SA1

Действие	Программа
Проверка отказа постоянной единицы (*stuck-at-1 fault*)	clrf X movf X,0 addlw 0x0f0 btfsc STATUS,C return swapf X,0 addlw 0x0f0 btfsc STATUS,C return

Диагностика отказов изменилась для двух бит аккумулятора и для отказов *SA*1, несмотря на то, что в проверке используется только один полубайт (4 бита), а в последующем в других проверках все биты регистра. В данном случае это объясняется двумя обстоятельствами. Во-первых, последующая проверка на отказ *SA*0 проверяет только переходы из 1 в 0, поэтому она чувствительна для *SA*1, но никак не зависит от *SA*0. Во-вторых, проверка на $B(W)$ использует половину бит как 0, а вторую как 1. Соответственно, отказы $SA(W_0)$ и $SA(W_2)$ были распознаны как отказы монтажного И/ИЛИ.

С точки зрения взаимной проверки аксиоматических базисов изменение программы привело систему в такое состояние, которое показано на рисунке 2.

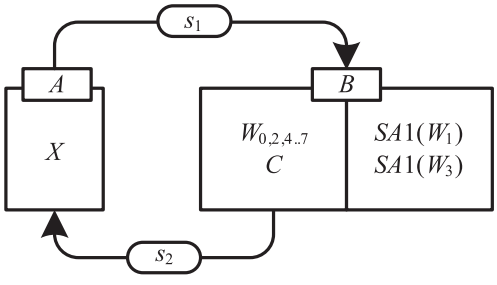

Рис.2. Усиление базиса для повышения отказоустойчивости

Функция проверки s_2 стала опираться на меньший базис (более сильный) и это позволило стать системе иммунитентной к соответствующим отказам. Таким образом, усиление базисов позволяет повышать отказоустойчивость систем, что говорит о том, что при разработке следует выбирать как можно более сильный базис, который меньше всего подвержен отказам.

Следует отметить, что усиление базисов не повлияло на безопасность системы при наличии отказов. Таким образом, описанным способом можно повышать отказоустойчивость систем с сохранением их безопасности.

Разработка и испытания отказоустойчивой системы

На практике функциональность системы требует большего числа ресурсов (памяти, устройств ввода-вывода и др.), чем операции по проверке базисов. В связи с этим усиление базисов для повышения отказоустойчивости является сложной и не всегда выполнимой задачей. В такой ситуации с точки зрения аксиоматических базисов для повышения отказоустойчивости возможно разделение задач безопасности и отказоустойчивости, и последующая их диверсификация, что показано на рисунке 3.

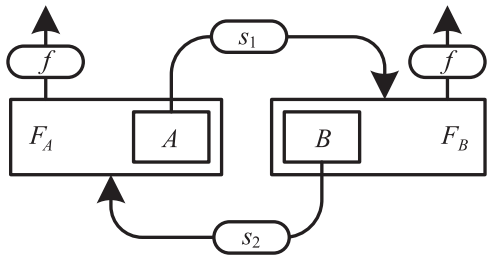

Рис.3. Отказоустойчивая диверситетная система

Здесь рассматривается четыре базиса A, B, F_A и F_B. Базис A сильнее базиса F_A (нарушение базиса A всегда ведет к нарушению F_A) и на нем проверяется базис F_B с помощью функции s_1 (аналогично для базисов B, F_B, A и функции s_2). Две реализации функции f базируются на независимых друг от друга базисах F_A и F_B. Если проверка базиса происходит успешно (например, результат s_1), то функция f, основанная на соответствующим базисе (F_B), может быть безопасно выполнена.

Для описанной реализации отказы, нарушающие базис A или B, приводят систему в безопасное состояние, а при нарушении F_A или F_B при выполнении A и B сохраняется свойство отказоустойчивости.

Для апробации описанного способа повышения отказоустойчивости была доработана уже испытанная безопасная система (испытания 1), в которую была добавлена функциональность, заключающаяся в генерации цифрового сигнала из микроконтроллера в виде последовательности нулей и единиц (01010101 и 11101000 соответственно). Т.е., ПО, работая по внутреннему циклу, на каждом из них изменяет выходной сигнал на 0 или 1 (согласно заданной последовательности). Для этого в памяти хранится слово из одного байта, над которым выполняется циклический сдвиг и старший бит отправляется на внешний порт. Работа системы во время отсутствия отказов показана на рисунке 4.

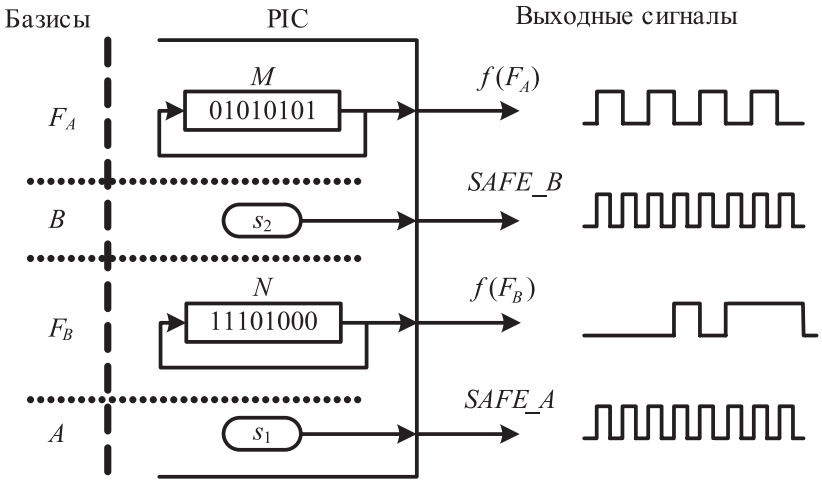

Рис.4. Работа системы во время отсутствия отказов

Для описанной системы была выполнена программа испытаний, результаты которой представлены в таблице 6. Испытания показали, что система сохраняет свойство безопасности поведения при наличии отказов (в случае отказа соответствующий базису сигнал *SAFE_A/SAFE_B* переходит в состояние 0).

Вторым результатом является полученное свойство отказоустойчивости. Если *SAFE_A* имеет постоянное значение 0, а *SAFE_B* переменный (~), то $f(F_A)$ всегда выполняется. Аналогично, если *SAFE_B*=0, а *SAFE_A* есть (~), то выполняется $f(F_B)$.

***Таблица 5* – Алгоритм отказоустойчивой системы**

№	Базис	Действие
1	*B*	Проверка *X*
2	F_A	Копирование из *M* в *X*
3	*B*	Проверка *M*
4	F_A	Копирование из *X* в *M*
5	F_A	Выполнение функции *f*
6	*A*	Проверка *W*, *C*
7	F_B	Копирование из *N* в *W*
8	*A*	Проверка *N*
9	F_B	Копирование из *W* в *N*
10	F_B	Выполнение функции *f*
11		Переход к пункту 1

Для реализации описанной функциональности в каждом из базисов требуется дополнительный байт памяти, который может быть подвержен отказам (байт *M* для базиса *A*, и байт *N* для базиса *B*). Последовательность действий алгоритма, на каких базисах они основывались и описание показаны в таблице 5.

Распределение операций между базисами следует из особенностей проверки базисов и используемых ресурсов. Так как проверка базиса подразумевает изменение ячеек памяти, а их содержимое требуется сохранять для выполнения задачи, то во время проверки требуется копирование, которое выполняется на другом базисе, нежели проверка. Все действия, касающиеся вычисления функции *f*, реализованы на соответствующем базисе (F_A или F_B), что обеспечивает отказоустойчивость одного из них в случае нарушения другого.

Необходимость распределения операций между базисами можно рассмотреть в виде правила, согласно которому для служебных операций во время подготовки процедур проверки требуется использовать базис, подлежащий проверке. Например, проверка F_A делается на основании *B*, и при этом все служебные операции требуется выполнять на F_A.

Заключение

Выбор базисов и разработка процедур проверки показали на практике, что рекомендуется усиливать базис насколько это возможно, что положительно отражается

Таблица 6 – Результаты испытаний отказоустойчивой системы

Отказы		Результаты			
Тип	Кол-во	Безопасность		Корректность исполняемой функции	
		SAFE_A	SAFE_B	$f(F_A)$	$f(F_B)$
–	1	~	~	+	+
$SA1(W)$	8	0	0	–+–+–+–+	––––+–+++
$SA0(W)$	8	0	0	+–+–+–+–	+++–+–––
$SA1(X)$	8	0	0	–+++++++	+++++++++
$SA0(X)$	8	0	0	–+++++++	+++++++++
$SA(C)$	2	0	0	++	––
$B_{OR}(W)$	7	0	0	–––––––	++–––++
$B_{AND}(W)$	7	0	0	–––––––	++–––++
$B_{OR}(X)$	7	0	0	–++++++	+++++++
$B_{AND}(X)$	7	0	0	–++++++	+++++++
$B_{OR}(C)$	1	0	~	+	–
$B_{AND}(C)$	1	0	~	+	–
$SA1(M)$	8	~	0	––––––––	+++++++++
$SA0(M)$	8	~	0	––––––––	+++++++++
$SA1(N)$	8	0	~	+++++++++	––––––––
$SA0(N)$	8	0	~	+++++++++	––––––––
$B_{OR}(M)$	7	~	0	–––––––	+++++++
$B_{AND}(M)$	7	~	0	–––––––	+++++++
$B_{OR}(N)$	7	0	~	+++++++	–––––––
$B_{AND}(N)$	7	0	~	+++++++	–––––––

Примечание: B_{OR} и B_{AND} – отказы монтажного ИЛИ и И соответственно; (+/–) – успешное или неуспешное выполнение функции; Позиция (+/–) от младшего бита регистра к старшему.

как на безопасности, так и отказоустойчивости системы. В случае задействования большого числа элементов, подверженных большому числу отказов, усложняются процедуры проверки базисов. Следовательно, при проектировании безопасной или отказоустойчивой микропроцессорной системы важной задачей является получение некоторого уже проверенного небольшого (т.е. сильного) базиса. Такой базис должен обладать значительным уровнем универсальности применения и на его основании можно в дальнейшем проверять отличную от базиса функциональность.

Таким образом, проведена апробация метода взаимной проверки аксиоматических базисов, который позволяет формализованно разрабатывать и верифицировать безопасные и отказоустойчивые системы, способные обнаруживать отказы и, как следствие, переходить в безопасное состояние или диагностировать проблему с сохранением работоспособности. Результаты испытаний показывают, что с помощью метода взаимной проверки аксиоматических базисов можно выйти на новый уровень формализации и качества в разработке и верификации отказоустойчивых и безопасных микропроцессорных систем.

Литература

1. **Бочков К. А.** Микропроцессорные системы автоматики на железнодорожном транспорте: учеб. пособие / К. А. Бочков, А. Н. Коврига, С. Н. Харлап // Гомель, БелГУТ. – 2013.

2. **Сивко Б. В.** Аксиоматико-базисный подход для разработки безопасных и отказоустойчивых систем / Б. В. Сивко // Автоматика на транспорте: ПГУПС. – 2015. – № 4, т. 1., С. 381–399.

3. **Бочков К. А.** Методы и средства доказательства функциональной безопасности микроэлектронных систем железнодорожной автоматики / К. А. Бочков, С. Н. Харлап, Д. Н. Шевченко // Електромагнітна сумісність та безпека на залізничному транспорті. – 2011. – № 2. – С. 73–81.

4. **Сивко Б. В.** Диверситетные аксиоматические базисы для разработки безопасных и отказоустойчивых систем / Б. В. Сивко // Вестник БелГУТа: Наука и Транспорт. – 2014. – № 1(28). – С. 19–23.

5. **Bates M**. PIC microcontrollers: an introduction to microelectronics. / M. Bates // Elsevier. – 2012. – 441 p.