РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
А В Т О М Е Т Р И Я
№ 5 1999

СИСТЕМЫ ОБРАБОТКИ ДАННЫХ

УДК 681.326 : 681.324

Ю. К. Димитриев
(Новосибирск)

ОБ ОДНОМ МЕТОДЕ СИНТЕЗА ТЕСТА ДЛЯ ЖИВУЧИХ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Рассматривается метод синтеза псевдослучайного теста для оперативного определения технического состояния самодиагностируемых живучих распределенных вычислительных систем.

Введение. Современный уровень развития распределенных вычислительных систем (ВС) на базе микропроцессорных СБИС характеризуется использованием средств тестовой проверки системы в сочетании со встроенными аппаратурными средствами (обычно вводимыми для проверки лишь интенсивно используемых узлов и информационных трактов) или без них.

Построение масштабируемых живучих распределенных мультимикропроцессорных систем (ниже – микроВС) требует автоматизировать определение их технического состояния (самодиагностируемые ВС) при наличии множественных отказов, когда единицей диагностирования является целый микропроцессор (МП). В [1] для ВС со связями типа «точка–точка» между МП предложены подход и основанный на нем децентрализованный алгоритм, которые обеспечивают управление взаимотестированием микропроцессоров и сопоставительный анализ результатов тестирования, ведущие к определению технического состояния системы (системное диагностирование).

Оперативное системное диагностирование живучей ВС предполагает использование резидентных тестов. Использование псевдослучайного тестирования позволяет строить резидентные тесты, компактные по занимаемой ими оперативной памяти. Однако время выполнения псевдослучайного теста находится в прямой зависимости от числа команд тестируемого МП. В [2] для исчерпывающего тестирования МП в условиях его эксплуатации предложено использовать представительное множество его команд. Применение подхода [2] к системному диагностированию с помощью псевдослучайного тестирования позволяет уменьшить время определения состояния системы.

В данной работе рассматриваются вопросы синтеза псевдослучайного теста на основе развития подхода [2] для определения технического состояния живучих ВС.

Обоснование метода синтеза. Проблема синтеза теста для масштабируемых живучих ВС является открытой, несмотря на разработку и выпуск многих коммерческих отказоустойчивых систем. Имеющиеся наработки в создании тестов микроВС могут найти лишь ограниченное применение для масштабируемых живучих ВС. Это связано, во-первых, с тем, что средства определения технического состояния микроВС рассчитаны на обнаружение одиночных отказов. Во-вторых, определение технического состояния микроВС, даже таких мощных, как GC фирмы "Parsytec", осуществляется в неоперативном режиме с участием оператора системы, контролирующего исполнение теста. Достигаемый при этом уровень автоматизации процедуры определения технического состояния ВС неприемлем для живучих масштабируемых систем. В последних управление тестированием должно основываться на использовании программ имитации поведения оператора и предусматривать его вмешательство в процесс самодиагностирования лишь в исключительных случаях (катастрофические отказы).

В-третьих, при неоперативном выполнении тест ВС не ограничен в использовании ресурсов системы, в частности времени и оперативной памяти (ОП), между тем как тест живучей ВС должен рассматриваться как пользовательская программа, для которой недопустима монополизация системы на неограниченное время. Далее, тест микроВС использует всю ОП системы, а последовательные его секции по мере необходимости могут подкачиваться в ОП из внешней памяти ВС. Тест же живучей ВС должен быть резидентен в ее оперативной памяти, а диагностирование должно осуществляться без разгрузки ОП.

Итак, резидентность и исполнение без разгрузки ОП являются определяющими условиями при разработке теста живучей ВС, наряду с требованиями минимизации затрат памяти для хранения теста (в том числе результатов его выполнения) и времени самодиагностирования.

Из-за сложности СБИС, используемых для построения ВС, и ограничений на возможность доступа к внутренним элементам СБИС сложившаяся практика проектирования средств диагностирования микроВС состоит в применении детерминированных функциональных тестов, когда в качестве примитивов при построении тестов выступают не аппаратурные узлы микропроцессора, а функции, реализуемые им при выполнении машинных команд.

Для разработки функционального теста МП требуется создать модель его неисправностей. От степени детализации модели зависит достоверность определения состояния МП. Основные недостатки метода функционального диагностирования – сложность разработки тестов, необходимость подбора операндов проверяемых команд и значительный объем ОП, занимаемой тестом. Разработчик аппаратуры, в состав которой входят микропроцессорные СБИС, как правило, не располагает ни достаточно детальным описанием МП, ни временем для разработки подобной модели.

Чтобы упростить для пользователя создание детерминированного функционального теста МП, сократить время выполнения теста и занимаемой им ОП, в работе [2] предложено осуществлять проверку некоторого представительного подмножества множества команд. Это подмножество образуют команды, которые в совокупности содержат все состояния устройства управления МП, необходимые для реализации функциональных преобразований данных, осуществляемых полным множеством команд. Описана общая методика выделения такого подмножества команд, называемого покрываю-

щим. Для сигнального процессора TMS32010, имеющего 79 команд с различными сложными способами адресации операндов, покрывающее множество состоит из 35 команд, в которых используются только простейшие способы адресации [2]. Однако получаемый для покрывающего множества команд детерминированный тест оказывается еще недостаточно компактным для оперативного диагностирования живучей ВС, а сама методика нуждается в детализации.

В качестве альтернативы использованию детерминированных функциональных тестов в [3] применяют генераторы псевдослучайных последовательностей (ГСП) для двоичных тестовых наборов. Эти тестовые наборы, подаваемые на информационные и управляющие входы МП, интерпретируются как коды команд МП и значения используемых ими операндов. Суть диагностирования заключается в сравнении результатов тестирования объектного МП с результатами тестирования эталонного при подаче на их информационные и управляющие входы одних и тех же тестовых наборов.

Основной недостаток диагностирования с применением ГСП состоит в том, что тестовые наборы могут не соответствовать семантически или синтаксически правильным кодам команд, а также представлять коды машинных команд или сочетания кодов команд, запрещенные к использованию. Для устранения этого недостатка в [4] двоичные наборы псевдослучайной последовательности рассматривают как ссылки на короткие детерминированные тесты – тестовые фрагменты. Каждый тестовый фрагмент предназначается для проверки группы родственных машинных команд или отдельного устройства МП. Для микропроцессора КР580ИК80А применение метода [4] позволило сократить память теста с 4500 до 2800 слов. Недостаток подхода [4] состоит в необходимости разрабатывать и хранить в ОП тестовые фрагменты, обеспечивающие проверку полного множества команд машинного языка.

Описанные в [3, 4] реализации обеспечивают проверку МП в неоперативном режиме с помощью диагностического комплекса, включающего аппаратурно-реализованный ГСП, эталонный и проверяемый МП, схему сравнения результатов тестирования микропроцессоров и управляющую ЭВМ. Последняя инициирует работу ГСП, синхронизирует функционирование эталонного и проверяемого МП, оценивает результаты тестирования этих МП.

Метод синтеза и выполнение теста ВС. Наш подход заключается в объединении методов [2–4] таким образом, что тест представляет собой псевдослучайную последовательность тестовых фрагментов, каждый из которых обеспечивает проверку одной машинной команды из покрывающего множества. В связи с этим в дальнейшем тестовые фрагменты будем называть эталонами команд (ЭК). Каждый ЭК содержит проверяемую команду и, возможно, команды подготовки ее операндов.

При необходимости в качестве ЭК могут использоваться также тестовые фрагменты для проверки функциональных блоков МП (например, оперативной памяти) или некоторых сочетаний машинных команд. Для проверки выполнения команд при экстремальных значениях операндов вводятся отдельные ЭК.

Использование ЭК уменьшает объем занимаемой тестом ОП, а употребление покрывающего множества команд сокращает время тестирования. Существенным фактором сокращения ОП, занимаемой тестом, и времени его выполнения является укрупнение единицы диагностирования до целого МП.

Это позволяет для диагностирования ВС использовать более простые контролирующие (без идентификации отказавшего блока) тесты МП.

Оперативное самодиагностирование живучей ВС осуществляется следующим образом. При наступлении событий, требующих определения состояния ВС, каждый МП вступает во взаимное тестирование с соседними микропроцессорами. Взаимное тестирование состоит в выполнении каждым из МП псевдослучайного теста, обмене результатами тестирования с соседними МП и сравнении этих результатов.

Формируемая псевдослучайная последовательность ЭК делится на подпоследовательности, каждая из которых выступает как секция детерминированного теста.

Обмен результатами тестирования между парами МП производится после выполнения очередной секции теста. Двоичный результат сравнения, выполняемого каждым МП пары, рассматривается им как оценка совпадения или несовпадения собственного технического состояния с техническим состоянием соседнего МП. Выполнение теста прекращается, если результаты очередной секции теста не совпали хотя бы в одном МП пары. Таким образом, деление теста на секции уменьшает время тестирования ВС.

Последующий сопоставительный анализ оценок, полученных для всех пар соседних МП, позволяет определить техническое состояние системы в целом, исходя из условия, что число отказавших МП не превышает заданного предельного значения t [1]. Таким образом, роль эталонного для каждого МП системы выполняет соседний с ним микропроцессор. Децентрализованное адаптивное управление взаимным тестированием и сопоставительным анализом его результатов производится программно-аппаратурными средствами, входящими в состав каждого микропроцессора ВС.

Модель микропроцессора. Следуя [2], представляем МП в виде набора функциональных блоков и устройства управления ими. Для каждой команды устройство управления вырабатывает определенную последовательность микрокоманд. Микрокоманда состоит из непустого множества микроопераций. При выполнении микроопераций, поступающих от устройства управления, функциональные блоки осуществляют требуемые преобразования информации. Функциональными блоками являются регистры, шины, комбинационные схемы различного назначения или целые узлы МП, такие, как арифметико-логическое устройство.

Всякое функциональное преобразование информации рассматриваем как последовательность микроопераций функционального преобразования. Базовая конструкция микрооперации преобразования – двуместная операция вида

$$R[n \ldots m] \leftarrow P[n \ldots m] * Q[n \ldots m],$$

где R, P, Q – используемые элементы памяти, шины, входы или выходы функциональных преобразователей, называемые регистрами; P и Q – регистры операндов; R – регистр результата; $[n \ldots m]$ – разряды используемых регистров (читается: «начиная с разряда n и кончая разрядом m», $n > m$); $*$ – операция выполняемого функционального преобразования. Частным случаем микрооперации преобразования является одноместная операция

$$R[n \ldots m] \leftarrow C_i[n \ldots m]$$

установки регистра R в фиксированное состояние C_i, например начальное; здесь C_i – воображаемый регистр, сопоставленный с выходами комбинационной схемы, которая вырабатывает сигнал установки для R.

В разных командах одна и та же микрооперация преобразования или последовательность таких микроопераций могут исполняться безусловно или по условию, в качестве которого используются значения булевых функций, вырабатываемых с помощью микроопераций условия

$$\alpha \leftarrow f_i(x_1, x_2, ..., x_m),$$

где $x_1, ..., x_m$ – логические переменные, взятые с чертой или без черты; f_i – булева функция; α – уникальное имя микрооперации (выхода соответствующей комбинационной схемы). Примером микрооперации условия служит микрооперация проверки на равенство нулю содержимого m-разрядного регистра R:

$$\alpha = \wedge_{i=1}^{m} \overline{R[i]}.$$

Для исчерпывающего тестирования МП необходимо проверить реализацию каждой микрооперации преобразования и каждой микрооперации условия. Множество команд, содержащих в совокупности все микрооперации, входящие в состав полного множества машинных команд, называют покрывающим множеством (ПМК).

В условиях эксплуатации единственным источником сведений о функциональных блоках МП и выполняемых ими преобразованиях информации служит техническое описание МП. В некоторых случаях сведения о функциональных блоках могут быть неполными. Это увеличивает избыточность ПМК, но не сказывается на применимости самого метода выделения ПМК и не влияет на диагностические возможности, когда единицей диагностики является целый МП.

Используемая модель неисправностей охватывает одиночные и множественные константные неисправности микропроцессора. Неисправность интерпретируется как неверное выполнение выделенных элементарных преобразований.

Конструкция матрицы представления команд МП. Основным рабочим элементом при выделении ПМК является матрица $M = \|m_{ij}\|$. Строки матрицы соответствуют командам МП, а столбцы – микрооперациям преобразования и условия. Элемент m_{ij} матрицы принимает значение «+» или «∅» соответственно тому, используется или нет j-я микрооперация в i-й команде.

Если среди выполняемых МП микроопераций имеются некоторые микрооперации преобразования, выполняемые безусловно и условно, то при построении ПМК возможны две стратегии: точная и грубая. При использовании точной стратегии одна и та же микрооперация преобразования, выполняемая по условию и безусловно, рассматривается как две разные микрооперации. При грубой стратегии микрооперация, выполняемая по условию, поглощает безусловную. Поглощаемая микрооперация в матрицу M не включается. Использование принципа поглощения упрощает описание МП, но уменьшает наблюдаемость его состояний.

Матрица M имеет большую размерность. Для облегчения работы ее разделяют по горизонтали на n подматриц: $M = \{M_i,\ i = 1, 2, ..., n\}$, содержащих родственные команды (например, команды пересылки информации, ариф-

метические, передачи управления и т. п.). Покрывающее множество команд определяют отдельно для каждой подматрицы, после чего подматрицы объединяют и для объединенных матриц находят новое ПМК. Этот процесс повторяют, пока не будет построено окончательное ПМК.

Заметим, что увеличение числа подматриц упрощает построение ПМК, но при этом растет избыточность покрывающего множества команд.

Алгоритм построения ПМК. Первый (неформализуемый) этап процедуры образования ПМК состоит в конструировании матрицы M. На этом этапе, исходя из имеющихся описания команд и функциональной схемы МП, выделяют набор микроопераций, вырабатываемых устройством управления, и производят разбивку матрицы M на подматрицы M_i. На втором этапе выполняют итеративную процедуру определения ПМК для подматриц M_i и их объединений.

Используемый нами способ построения ПМК отличается от описанного в [2] следующим:

1. Применяется булева матрица представления команд МП, что упрощает выделение ПМК.

2. Для представления команд использован язык регистровых передач с дифференциацией микроопераций на условные и безусловные, что позволяет повысить наблюдаемость состояний МП.

3. При построении ПМК использованы оценки сложности команд, динамически изменяющиеся в процессе формирования ПМК.

Назовем существенными i-ю микрооперацию, если она используется в единственной j-й команде, и саму эту команду. В матрице M существенной микрооперации соответствует столбец, имеющий знак «+» в единственной строке. Существенность команды является достаточным условием ее принадлежности к ПМК.

Весом w_j команды j называем число знаков «+» в j-й строке матрицы M.

Для выделения ПМК выполняются следующие процедуры.

Процедура 1. Пусть M_i – некоторая подматрица матрицы представления МП, описывающая множество родственных команд и содержащая p_j строк, а M_i^k – формируемое для нее ПМК. В исходном состоянии $M_i^k = \varnothing$.

1. Пока в M_i имеются существенные команды, для $j = 1, 2, \ldots, p_j$ выполнять: если j-я строка M_i существенна, то выполнить п. 1.1.

1.1. Включить строку j в M_i^k, вычеркнуть из M_i строку j и столбцы, для которых элементы строки j не пусты.

2. Если остаточная матрица M_i не пуста, то перейти к п. 3, иначе обозначить M_i^n – сформированную матрицу вида M_i^k. Конец процедуры.

3. Для каждой строки остаточной матрицы M_i вычислить вес w_j, выбрать строку j с максимальным весом, выполнить п. 1.1 и затем перейти к п. 1.

Процедура 2. Пусть исходная матрица $M = \{M_i,\ i = 1, 2, \ldots, n\}$. Для включения в M выбираем команды, использующие простейшие возможные для них способы адресации операндов. (Простота адресации находится в обратной зависимости от числа требуемых для ее осуществления циклов обращения к памяти.)

1. Найти матрицу ПМК для каждой M_i с помощью процедуры 1.

2. Объединить попарно и произвольно матрицы M_i^n, полученные на шаге 1, образовав тем самым новое множество M.

3. Выполнять пп. 1 и 2, пока не будет получено одноэлементное множество матриц M_i. Найти для нее M_f^n, используя процедуру 1.

4. Каждый из возможных способов адресации использовать в одной произвольно выбираемой команде из M_f^n, допускающих такой способ. Ввести в M_f^n команды, которые не входят в нее, имеют в M минимальный вес и реализуют такие способы адресации, которые не используются в командах из M_f^n. Конец процедуры.

Пример матрицы M для гипотетического МП, выполняющего семь команд, реализующих в совокупности 10 микроопераций, приведен в таблице.

Для примера в таблице ПМК состоит из двух команд, перечисленных в порядке включения в него: $M_f^n = \{2,1\}$. Команда 2 является существенной по микрооперации 4 и включается в ПМК на шаге I. После вычеркивания строки 2 и столбцов 2, 4–9 образуется остаточная матрица, имеющая три столбца (1-й, 3-й и 10-й) и шесть строк. Команды 1, 3, 5 и 6 в остаточной матрице имеют одинаковый и максимальный вес. Для включения в ПМК произвольно выбрана микрооперация 1.

Описанная методика использована для выделения ПМК одноплатной микроЭВМ ряда «Электроника 60». Эта микроЭВМ реализует 107 команд с 12 различными способами адресации. Общее число реализуемых микроопераций 243.

Исходная матрица команд разделена на четыре группы. Первую группу составляют одноадресные команды (38 команд), вторую – двухадресные команды (18 команд), третью группу образуют команды управления (26 команд), а в четвертую группу вошли 25 команд расширенной арифметики и операций с плавающей запятой.

Для построения ПМК использована точная стратегия.

ПМК для команд первой группы состоит из 10 команд: семь команд обеспечивают покрытие множества безусловных микроопераций преобразования и три добавлены по микрооперациям условия.

ПМК для команд второй группы образовано пятью командами, выбранными по условиям покрытия безусловных микроопераций преобразования.

Код команды	Микрооперации										Вес команды по итерациям	
	1	2	3	4	5	6	7	8	9	10	I	II
1	+	+	+			+		+	+	+	7	3
2		+		+	+	+	+	+	+		7	0
3	+		+			+	+	+	+	+	7	3
4	+		+		+	+	+		+		6	2
5	+		+				+	+		+	5	3
6	+	+	+					+	+	+	6	3
7			+			+	+	+		+	5	2

В эти команды вошли все микрооперации условия, используемые в командах этой группы.

В ПМК третьей группы команд вошли две команды; используемые в них условия покрывают условия, применяемые в остальных командах группы.

Наконец, для четвертой группы ПМК состоит из четырех команд, покрывающих множество микроопераций преобразования, и еще шесть команд добавлены по микрооперациям условия.

На втором этапе для сводной матрицы, образованной из ПМК групп, выделено финальное ПМК, содержащее 33 команды. Все команды этого ПМК реализуются с использованием простейшего (регистрового) метода адресации. Для трех самых простых команд ПМК применены три более сложных способа адресации, при осуществлении которых не используются микрооперации, применяемые при регистровой адресации.

Программа генерации псевдослучайного теста. Синтез псевдослучайного теста МП и управление его исполнением осуществляются в оперативном режиме по условиям вызова самодиагностирования ВС. Синтез теста выполняет резидентная в ОП программа генерации. Исходными данными для нее служат структурированный массив MET эталонов команд и структурированный массив VET, называемый вектором эталонов. Эталон команды представляет собой семантически и синтаксически корректный код команды из ПМК (возможно, в сочетании с командами подготовки операндов). Эталоны команд оформлены как подпрограммы и могут использоваться независимо. С каждым элементом массива эталонов MET сопоставлен элемент VET, содержащий ссылку на этот ЭК и значение F_i функции распределения частоты его использования.

Пусть MET – упорядоченное множество эталонов команд: MET $=\{ЭК_1, ЭК_2, ..., ЭК_i, ..., ЭК_p\}$. Значение i-го элемента VET, сопоставленного с $ЭК_i$, равно $F_i = \sum_{j=1}^{i} f_j$, где f_i – статистическая вероятность использования $ЭК_j$; $\sum_{j=1}^{p} f_j = 1$.

При выполнении программы генерации теста (ПГТ) осуществляются следующие функции: 1) формирование и исполнение псевдослучайной последовательности ЭК; 2) сравнение полученных (промежуточных) результатов теста с эталонными значениями; 3) контроль за достижением условия завершения теста. Рассмотрим реализацию этих функций.

Выбор очередного ЭК производится с помощью генератора псевдослучайных чисел, равномерно распределенных в интервале от 0 до 1. Очередное значение случайной величины r, полученное от ГСП, последовательно сравнивается со значениями F_i вектора VET. Выполняется тот $ЭК_i$, для которого в VET имеет место $F_{i-1} \le r < F_i$, $p - 1 \ge i \ge 2$.

Тест завершается (функция 3) после выполнения заданного числа проходов ПГТ. Это число определяется экспериментально при инициализации программы генерации теста как минимальная длина псевдослучайной последовательности, обеспечивающая формирование заданной функции распределения частоты использования эталонов команд.

Время тестирования и достоверность определения состояния МП зависят от способа реализации функции 2. Рассмотрим эти способы подробнее.

Выбор метода формирования результатов. Согласно изложенному выше, в качестве эталонных для каждого модуля ВС используются результаты аналогичного теста, выполняемого смежным с ним МП. Для сравнения

собственных результатов с эталонными необходимо хранить результаты тестирования в оперативной памяти МП и передавать их соседним микропроцессорам. Обозначим через Y множество результатов псевдослучайного теста, используемых для сравнения с аналогичной информацией, полученной смежным МП системы. Элементами Y является содержимое программно-доступных регистров и ячеек оперативной памяти МП, объединяемых понятием памяти результатов (ПР).

Очевидный способ образования ПР Y состоит в том, чтобы отнести к ней все программно-доступные элементы МП (включая ячейки ОП), используемые для хранения результатов каждой из выполняемых команд теста. При реализации этого способа, называемого способом непосредственного полного сравнения, сопоставление результатов теста с эталонами осуществляется после выполнения каждой очередной команды теста.

Сравнение большого количества результатов дает высокую вероятность того, что несовпадение технических состояний взаимотестируемых МП будет замечено, даже если взаимотестируемые МП окажутся неисправными. Однако время тестирования при этом становится значительным из-за необходимости переключения от исполнения собственно теста к программе сравнения (и обратно) после осуществления каждого очередного эталона команды. Для такого переключения необходимо иметь встроенную аппаратуру. Для систем на базе серийных СБИС возможность введения такой аппаратуры исключена. Поэтому если иметь в виду программную реализацию рассматриваемой функции, то единственно возможным способом ее осуществления является выполнение сравнения после завершения группы последовательных команд теста. Это сокращает время тестирования, но требует использования для временного хранения подлежащих сравнению данных структурно выделенной для этого области ОП, что крайне нежелательно при оперативном самодиагностировании ВС.

Можно получить приемлемые значения времени тестирования без сопутствующего значительного увеличения занимаемой тестом ОП, если использовать метод непосредственного выборочного сравнения результатов теста, выполняемого взаимотестируемыми МП. При реализации этого способа в множество Y включаются результаты определенных последовательностей ЭК. После заполнения ПР (соответствующее множество ЭК составляет секцию теста) находящуюся в ней информацию сравнивают с информацией из ПР соседнего МП. Чтобы обеспечить наблюдаемость всех состояний тестируемого МП, в множество Y следует включить результат каждой из независимых по обрабатываемой информации последовательностей ЭК (достаточное условие). Наиболее просто это требование удовлетворяется, если весь тест представляет собой единую цепь информационно связанных ЭК. Тогда становится возможным использовать для формирования Y регулярные правила, например, такие, как включение в Y результата каждого $(c \times k)$-го ЭК, где c – константа и $k = 1, 2, \ldots$. В следующем ниже изложении имеется в виду именно этот случай. Таким образом, после c проходов ПГТ результат выполнения очередного ЭК записывается в очередную ячейку ПР тестируемого процессора.

Одно из основных требований к псевдослучайному тесту, реализуемому в оперативном режиме без разгрузки ОП, состоит в ограничении памяти, выделяемой для хранения операндов и результатов тестирования. Поэтому в качестве операндов для ЭК следует использовать содержимое памяти результатов, которая включает в себя регистры общего назначения МП, и немодифи-

цируемые части и исходные данные самой программы генерации теста (включая подпрограммы реализации эталонов команд).

Применение случайных операндов (операнды первого вида) позволяет реализовать исчерпывающую проверку заданной команды, что достигается использованием достаточно большого числа реализаций соответствующего ЭК (длины формируемой псевдослучайной последовательности).

Подбирая детерминированные операнды (операнды второго вида), можно с помощью коротких псевдослучайных последовательностей (следовательно, за небольшое время) проверить выделенные состояния команды (связанные, например, с получением нулевого результата или результата с заданным знаком). Однако сокращение времени тестирования за счет проверки команд на детерминированных операндах оплачивается потерей полноты проверки команды. (Заметим, однако, что неполнота проверки – обычное свойство детерминированных тестов; она считается оправдываемой малым временем тестирования.) Разумный компромисс между полнотой и временем тестирования может быть достигнут для псевдослучайного теста включением в ЭК команд, которые работают и со случайными, и с детерминированными (выделенными) операндами.

Метод непосредственного выборочного сравнения требует меньше ОП, чем метод непосредственного полного сравнения, однако не решает проблемы памяти полностью.

Кардинальное решение этой проблемы состоит в использовании сжатия информации на основе формирования сигнатур [5] в применении к методу непосредственного выборочного сравнения. Сжатие результатов тестирования, достигаемое применением сигнатур, не только сокращает затраты ОП на хранение теста, но и уменьшает время тестирования за счет уменьшения объема результатов тестирования, передаваемых между МП.

Пусть емкость памяти результатов, используемой в ПГТ, составляет n ячеек. При выборочном сравнении общая длина подлежащей сжатию выходной последовательности, образуемой содержимым ПР, составляет $L = Qnm$ бит, где Q – число циклов формирования содержимого ПР; m – разрядность машинного слова. Будем рассматривать ПР как многовыходную цифровую схему. Сопоставим отдельный выход цифровой схемы с каждой ячейкой ПР. Тогда на каждом из n выходов формируется выходная последовательность длиной $l = Qm$ бит*. Простой метод образования сигнатуры для многовыходной цифровой схемы, имеющий также несложную программную реализацию, описан в [5] и заключается в использовании следующего двухступенчатого преобразования. Первая ступень преобразования состоит в слиянии n выходных двоичных последовательностей $y_i(k)$ длиной l в одну последовательность $y_0(k)$ в соответствии с правилом

$$y_0(k) = \oplus_{i=1}^{n} y_i(k), \qquad i = 1,2,\ldots,n, \quad k = 1,2,\ldots,l,$$

---

* Так как для всех методов сжатия значение l определяется выражением $l \le 2^m - 1$, то это дает основание рассчитать значение Q: $Q \le \lfloor (2^m - 1)/m \rfloor$. Заметим также, что число N циклов выполнения ПГТ связано со значением Q отношением $N = cQ$, где c – константа. Последнее соотношение можно использовать при выборе генератора псевдослучайных чисел, период которого должен удовлетворять неравенству $P \ge N$.

где $\oplus$ – операция суммирования по mod2. Такие преобразования выполняются после каждого очередного заполнения ПР. Тем самым в очередной q-й, $q=1,2,\ldots,Q$, секции теста формируется часть последовательности $y_0(k)$: от $y_0(m(q-1))$-го до $y_0(mq)$-го разряда. На второй ступени преобразования часть последовательности $y_0(k)$, полученная при выполнении секции q, и сигнатура последовательности $y_0(k)$, полученная при выполнении предшествующих $(q-1)$-й секций, сжимаются в m-разрядную сигнатуру $S(x)$ в соответствии с используемым порождающим полиномом. Если тест выполняется полностью, то подобное сжатие с последующим сравнением полученной сигнатуры с сигнатурой, образованной соседним микропроцессором ВС, выполняется Q раз.

Поскольку вероятность необнаружения кратных ошибок при использовании сигнатурного анализатора зависит от степени порождающего полинома, в качестве такового используют полином, имеющий для заданного m наибольшую степень. Построены каталоги оптимальных полиномов; так, для $m=16$ оптимальным является полином $\varphi(x)=1\oplus x^7\oplus x^9\oplus x^{12}\oplus x^{16}$.

Расчетные выражения для оценки достоверности рассматриваемого двухступенчатого преобразования даны в [5], исходя из предположений, что $l<2^m$ и события обнаружения ошибки в последовательностях $\{y_i(k)\}$ в результате анализа $\{y_0(k)\}$ и в последовательности $\{y_0(k)\}$ при использовании m-разрядной сигнатуры $S(x)$ являются независимыми. Тогда вероятность необнаружения ошибки кратности μ составляет:

$$P_{n0}^{\mu}=P_{n1}^{\mu}+\frac{1}{2^m}-P_{n1}^{\mu}\frac{1}{2^m},$$

где P_{n1}^{μ} – вероятность необнаружения μ-кратной ошибки на первой ступени преобразования:

$$P_{n1}^{\mu}=\begin{cases}0, & \text{если } \mu\neq 2k;\\ \cong\prod(2j-1)\left(\dfrac{n-1}{n}\right)^{\mu/2}\dfrac{1}{1^{\mu/2}}, & \text{если } \mu=2k \text{ и } \mu\ll l.\end{cases}$$

Здесь k – произвольное целое.

Заключение. Описан метод выделения представительного множества команд микропроцессоров и систем на их основе, использование которого позволяет синтезировать компактные функциональные тесты.

Применение этого метода к микроЭВМ ряда «Электроника 60М» позволило сократить набор команд, используемых для построения теста, со 107 до 33.

Метод выделения ПМК использован совместно с методом синтеза псевдослучайных тестов. Данный подход перспективен для синтеза теста самодиагностируемых живучих вычислительных систем. Описана программа генерации в оперативном режиме псевдослучайного теста с заданным законом распределения частоты использования команд. Примененные в программе генерации теста решения дают возможность достичь компромисса между занимаемой тестом памятью и временем его выполнения.

Важное преимущество предлагаемого подхода состоит в том, что его применение для оперативного диагностирования ВС позволяет формировать тест, состав команд которого и частота их использования адекватны задаче пользователя. Прогонка теста определяет пригодность МП для решения такой задачи. Возможность применять частично исправный МП для решения задачи пользователя повышает живучесть ВС. В сочетании с дублированием решения задачи пользователя на разных МП подобные тесты способны обнаруживать перемежающиеся отказы. Наконец, на основе данного метода можно синтезировать «смеси» команд для оценки эффективности ВС для решения конкретных классов задач. Синтез подобных тестов может быть автоматизирован.

Работа выполнена при поддержке Российского фонда фундаментальных исследований (гранты № 96-01-01520 и № 98-01-00402).

СПИСОК ЛИТЕРАТУРЫ

1. **Димитриев Ю. К.** Самодиагностика модульных вычислительных систем. Новосибирск: Наука, 1993.
2. **Talkhan E.-S. A., Ahmed A. M. N., Salama A. E.** Microprocessors functional testing techniques // IEEE Trans. on Comput. Aided Design. 1989. **8**, N 3. P. 316.
3. **Fedi X., David R.** Some experimental results from random testing of microprocessors // IEEE Trans. Instr. and Measure. 1987. **IM-35**, N 1. P. 78.
4. **Балюк В. В., Кобзар С. П., Куценко В. Н.** Применение генераторов псевдослучайных последовательностей для диагностирования микропроцессора К580 и К580А // Автоматика и вычисл. техника. 1989. № 3. С. 89.
5. **Ярмолик В. Н.** Контроль и диагностика цифровых узлов ЭВМ. Минск: Наука и техника, 1988.

Поступила в редакцию 25 декабря 1998 г.