УДК 621-192

С.Ф. ТЮРИН, А.А. ПРОХОРОВ

Пермская сельскохозяйственная академия им. академика Д.Н. Прянишникова, Россия

ОБЕСПЕЧЕНИЕ СБОЕУСТОЙЧИВОСТИ МИКРОПРОЦЕССОРНЫХ СИСТЕМ, АДАПТИРУЕМЫХ К ФУНКЦИОНАЛЬНЫМ ОТКАЗАМ НА ОСНОВЕ КОНТРОЛЯ ПРАВИЛЬНОСТИ ВЫПОЛНЕНИЯ КОМАНД ПЕРЕДАЧИ УПРАВЛЕНИЯ

В статье предлагается метод обеспечения сбоеустойчивости микропроцессорных систем на основе контроля правильности выполнения команд передачи управления по информации слова состояния программы.

сбоеустойчивость микропроцессорных систем, контроль, слово состояния программы

Введение

Повышение отказосбоеустойчивости цифровых элементов и узлов (ЦУ) и цифровой аппаратуры (ЦА) управления сложных технических комплексов, работающих в реальном масштабе времени, является одним из аспектов проблемы надежности и качества ее функционирования [1, 2]. Актуальность этой задачи возрастает по мере развития элементной базы, усложнения самой цифровой аппаратуры управления и внедрения ее во все отрасли национального хозяйства (авиацию, космонавтику, атомную энергетику и т.п.).

Существующие методы обеспечения отказосбоеустойчивости цифровой аппаратуры (ЦА) информационно-технологических систем, как правило, основаны на структурном резервировании, являющемся своего рода "внешним" для каждого из резервированных каналов и заключающемся во введении дополнительных копий ЦА, что обеспечивает с определенной вероятностью сохранение исходных функций. Это более чем в 3 раза увеличивает стоимость, массогабаритные показатели, энергопотребление. Кроме того, резервы сами "провоцируют" отказы, и такие методы эффективны на относительно небольшом временном интервале работы аппаратуры.

В последние годы поставлена задача разработки компьютеров высокой надежности, в которых рабочие, контрольные и восстановительные процессы составляют единое целое, которые могут функционировать без технического обслуживания и ремонта в течение всего срока эксплуатации, обладать способностью самовосстановления, адаптации к отказам и повреждениям, например, путем отключения пораженных участков и реализации требуемых функций на оставшемся количестве элементов с возможным допустимым замедлением скорости. Тем более, что уже созданы технологические предпосылки для создания таких высоконадежных "живучих" систем [3].

Так, в связи с широким распространением универсальных программируемых логических устройств (ПЛУ), которые могут реконфигурироваться, т.е. изменять функции элементов и связи между ними, в процессе эксплуатации возникают новые возможности для построения отказоустойчивых цифровых автоматов, как элементной базы этих новых систем. Предлагается создавать "живучие" информационно-технологические системы путем сохранения хотя бы базисных функций для заданной модели отказов, позволяющих вычислять исходные за большее время после соответствующей реконфигурации [4 – 7].

С целью обеспечения сбоеустойчивости, особенно актуальной при повышении уровня интеграции микросхем, наиболее целесообразным представляется сохранение резервированных (например, трои-

рованных) структур при отказах в катастрофических ситуациях путем перестройки их на основе сохранения универсальности технических комплексов, работающих в реальном масштабе времени, является одним из аспектов проблемы надежности и качества ее функционирования. **Цель работы** – разработка метода обеспечения сбоеустойчивости ЦА.

Метод и его реализация

Предложим модифицированный граф изменения состояний одноканальной микропроцессорной системы (МПС) с учетом возможности деградации функций в случае отказов в катастрофических условиях (рис. 1).

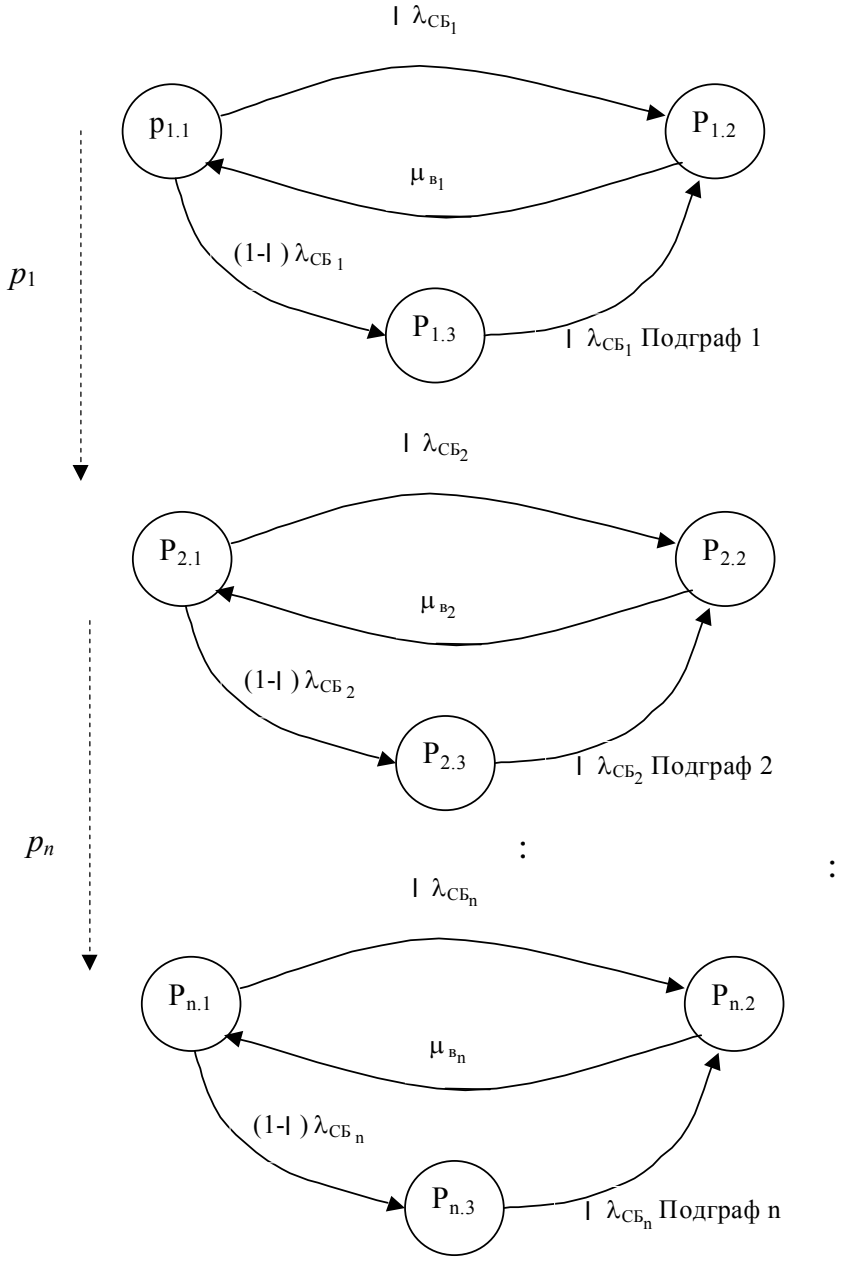

Рис.1. Модифицированный граф изменения состояний одноканальной МПС, состоящий из n подграфов, соответствующих *n*–1 уровню деградации производительности

Здесь $P_{i.1}$ – состояние отсутствия сбоев, встроенные средства контроля не фиксируют их наличие; $P_{i.2}$ – состояние наличия сбоя, когда и встроенные средства контроля их фиксируют; $P_{i.3}$ – состояние скрытого сбоя, т.е. незафиксированного средствами контроля; $\mu_Б$ – интенсивность восстановления; $\lambda_{сб}$ –

интенсивность сбоев; l – полнота контроля; i – уровень деградации производительности ЦВК, адаптируемого к функциональным отказам.

При этом принимаются следующие допущения, соответствующие положениям теории надежности:

– в состоянии, соответствующем исходному уровню производительности, система функционирует без отказов;

– переход на низший уровень производительности соответствует одному отказу.

Очевидно, что система может находиться только в одном из состояний.

Для повышения вероятности выполнения ЦА различных этапов работы необходимо обеспечить максимальную достоверность функционирования

$$D = \sum_{i=1}^{n} H_i (P_{i1} + P_{i2}) \to \max , \qquad (1)$$

где H_i – нормировочные вероятности нахождения ЦВК на i-м уровне деградации, 1 – исходный уровень с исходной производительностью, $i = \overline{1,n}$; P_1 – состояние отсутствия сбоев; P_2 – состояние обнаруженного сбоя.

Достижение этой цели будем стремиться путем увеличения полноты контроля l, что позволяет быстрее обнаружить сбой, приведший к нарушению передачи управления, а, значит, и увеличить интенсивность восстановления $\mu_в$.

В свою очередь, увеличение полноты контроля l может привести к увеличению интенсивности отказов λ (т.е. ускорить деградацию) и сбоев за счет введения дополнительной аппаратуры, увеличивает объем памяти и время выполнения команд, а процедуры восстановления не должны превышать заданных временных ограничений.

Возможны следующие варианты контроля по типу модификации программного обеспечения.

1. Использование одного типа программ – с контролем, тогда затраты памяти равны $W + W_k$, где W – исходный объем памяти, W_k – дополнительный объем памяти контрольных процедур, временные затраты, например, по максимальной длительности выполнения программы $T + T_k$, где T – исходная длительность, T_k – дополнительное время для контрольных процедур.

2. Использование программы без контроля в резервированной структуре и программ с контролем при работе в одноканальном варианте: $2W + W_k$, T, $T + T_k$.

3. Использование компиляции для введения контроля в случае перехода в одноканальную структуру с затратами памяти: $W_{комп}$, W, $W+W_{комп}+W_k$, T, $T_{комп}$, $T+T_{контр}$.

В случае аппаратной реализации контроля необходимо учесть дополнительные аппаратурные затраты A. Кроме того, учитываются стоимостные ограничения C.

Достоверность функционирования системы, описываемой графом 1, является функцией параметров графа (рис. 1):

$$D = f(1, \lambda_{сб_i}(t), \mu_{в_i}(t)). \qquad (2)$$

Непосредственную связь D, T, A, W в некотором одном выражении найти не представляется возможным, поэтому целесообразен вариантный синтез средств контроля с последующей Парето (Pareto) оптимизацией при заданных ограничениях $T_{доп}$, $A_{доп}$, $W_{доп}$, $C_{доп}$:

$$\text{Pareto}\ (D, \Delta T, \Delta A, \Delta W);$$

$$\lambda_{сб_i}(t) = \varphi(A + \Delta W), \quad \mu_{в_i}(t) = \psi(T + \Delta T);$$

$$D = \sum_{i=1}^{n} H_1 \left(\frac{1 + \dfrac{\lambda_{сб_i}}{\mu_{в_i}}}{\dfrac{1}{l} + \dfrac{\lambda_{сб_i}}{\mu_{в_i}}} \right). \qquad (3)$$

Рассмотрим новый предлагаемый подход к обеспечению сбоеустойчивости микропроцессоров [8].

В современных микропроцессорных системах на базе процессоров фирмы Intel все шире применяются средства контроля и диагностирования, введен контроль по нечетности шины адреса, информации и данных, контролируются операции в сопроцессоре, выполняется многоступенчатая конвейерная выборка команд, реализована «предсказание» направления ветвления.

Однако такой важный процесс, как контроль выполнения условного перехода по значению бита некоторого признака пока не реализован. Это значительно снижает достоверность функционирования и может привести к ошибкам вычислений.

Предлагаемое устройство (блок контроля) позволяет устранить этот недостаток путем получения возможности контроля условных переходов. Оно содержит блок памяти автоматной модели, триггеры логических условий, регистр управления записью признаков, регистр PSW (признаков), мультиплексор, дешифратор-демультиплексор, регистр состояния последующего, регистр кода программы, регистр сравнения и схему сравнения, логику записи информации в блок контроля и включается в типовую микропроцессорную систему (рис. 2).

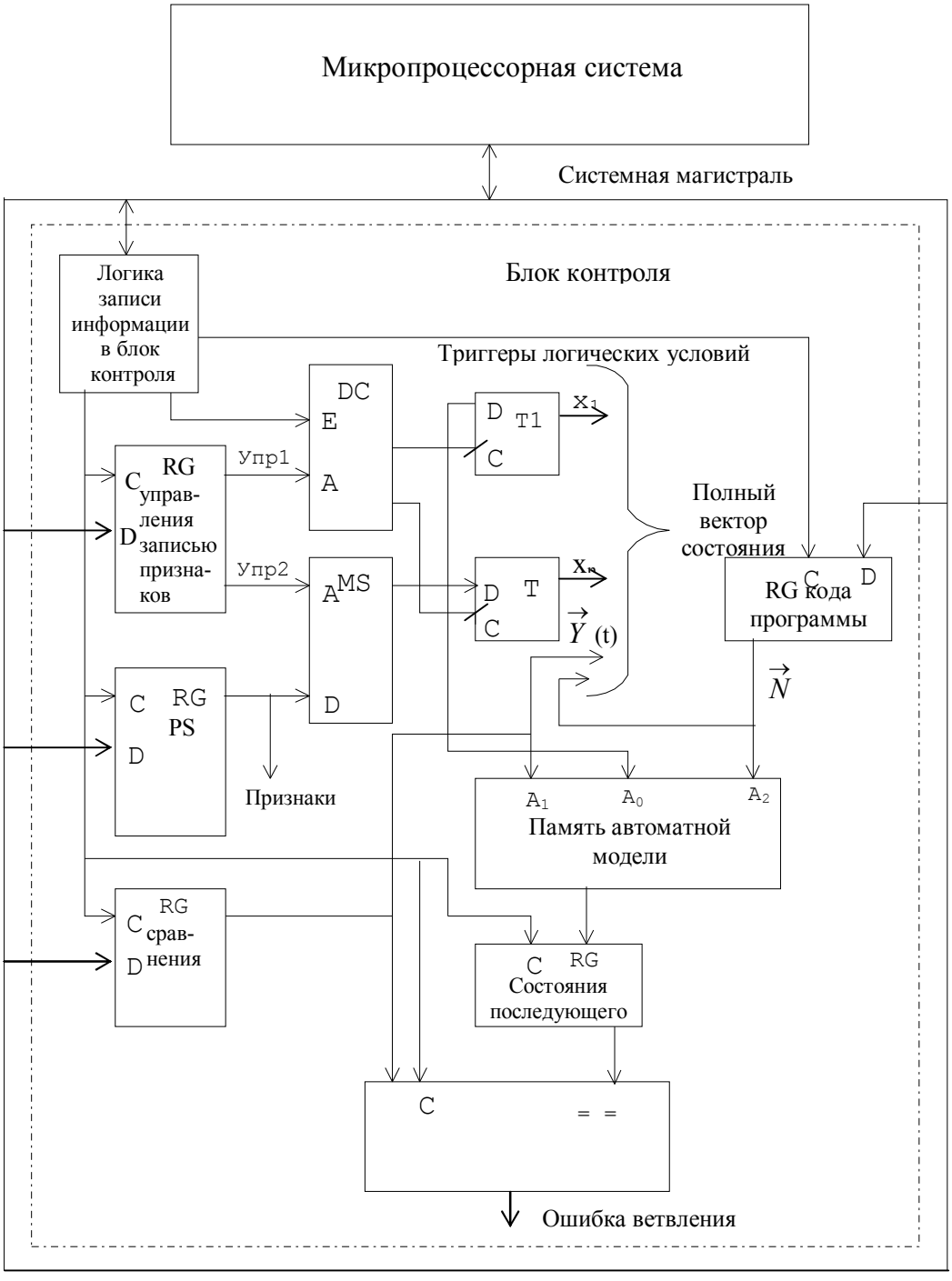

Рис. 2. Устройство контроля правильности выполнения команд переходов

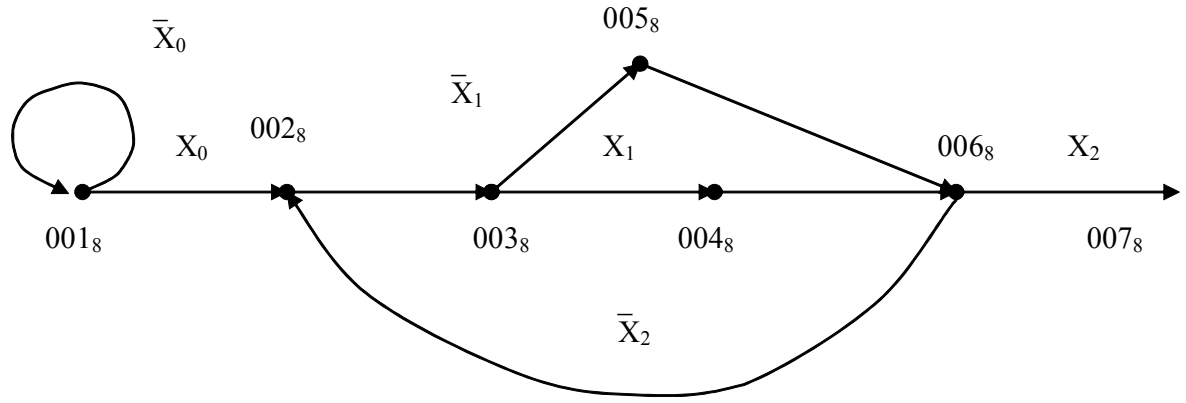

Рис. 3. Бинарный граф участка исполняемой программы

Пусть бинарный граф программы представлен на рис. 3. Условное распределение портов в восьмеричном коде: 200_8 – регистр сравнения (для записи номера выполняемого линейного участка); 300_8 – регистр управления записью признаков. Номера признаков: знак (S) – 111_2, ноль (z) – 110_2, вспомогательный перенос (AC) – 100_2, четность (P) – 010_2, перенос (C) – 000_2. Распределение разрядов информации в регистре управления записью признаков: номер признака – разряды 7, 6, 5; номер переменной – разряды 4, 3, 2, 1, 0.

Пусть граф исполняемой программы имеет вид рис. 3.

На графе (рис. 3) ребра – это линейные участки, отмеченные восьмеричными кодами своих номеров; дуги, помеченные значениями переменных, описывают ветвление программы. Тогда фрагменты некоторой программы управления с контролем условных переходов (Язык Ассемблер) могут быть описаны табл. 1. В табл. 1 обведены те команды, которые введены с целью контроля.

Таблица 1

Фрагменты программы управления с контролем условных переходов

Метка	Мнемокод	Комментарий
M1:	POP PSW	; восстановление значения признаков
M0:	MVI 001 OUT 200	; запись в A непосредственно операнда 001_8 ; установка регистра сравнения
		; линейный участок 001_8
	MVI 300 OUT 300	; подготовка к записи в регистр управления записью признаков: признак 110_2, x_0 ; установка регистра управления записью признаков
		; подготовка к операции
	SUB B	; определение A-B перед ветвлением
	PUSH PSW	; загрузка признаков в регистры PSW
	JNZ M1	; ветвление по признаку z
	POP PSW	; восстановление значения указателя стека
M6:	MVI 002 OUT 200	; фиксация прохождения второго линейного участка ; если равно нулю
		; линейный участок 002_8
	JMP M2	; безусловный переход на линейный участок 003_8
	MVI 003 OUT 200	; фиксация прохождения линейного участка 003_8
M2:	.	; линейный участок 003_8

Окончание табл. 1

	MVI 041 OUT 300	; подготовка к ветвлению: признак 010_2, x_1
	.	; подготовка к операции
	MOV A, C	; пересылка в A из C
	ORA A	; установка признаков
	PUSH PSW	; загрузка признаков в регистр PSW
	JPE M3	; переход по нечетному числу единиц
	POP PSW	; восстановление значения указателя стека
	MVI 004 OUT 200	; фиксация прохождения четвертого линейного ; участка, если четно
	⋮	; линейный участок 004_8
M5:	MVI 006 OUT 200	; фиксация прохождения шестого линейного участка
	.	; линейный участок 006_8
	MVI 342 OUT 300	; подготовка к ветвлению: признак 111_2, x_2
	.	
	MOV A, M	; загрузка в A из памяти
	ORA A	; установка признаков
	PUSH PSW	; загрузка признаков в регистры PSW
	JP M4	; переход по положительному результату
	POP PSW	; восстановление значения указателя стека
	MVI 007 OUT 200	; фиксация прохождения седьмого линейного участка
	.	; седьмой линейный участок
	MVI 000 OUT 200	; конец программы, обнуление регистра сравнения
	HLT	; останов
M3:	POP PSW	; восстановление значения указателя стека
	MVI 005 OUT 200	; фиксация прохождения пятого линейного участка
	.	; пятый линейный участок
	JMP M5	; безусловный переход на шестой линейный участок
M4:	POP PSW	; восстановление значения указателя стека
	JMP M6	; безусловный переход на метку M6 (линейный участок 0028)

Графу исполняемой программы (рис. 3) соответствует массив программирования блока памяти автоматной модели (табл. 2) с учетом нулевого содержимого регистра кода программы (разряда a_7, a_6).

Таблица 2

Массив программирования блока памятиавтоматной модели

№	Адрес								Данные			Комментарий
	A_2		A_1					A_0				
	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0	d_2	d_1	d_0	
0	0	0	0	0	0	0	0	0	0	0	1	Безусловный переход из 000 в 001
1	0	0	0	0	0	0	0	1	0	0	1	
2	0	0	0	0	0	0	1	0	0	0	1	Если $\bar{x}_0$, то 001 иначе 010
3	0	0	0	0	0	0	1	1	0	1	0	
4	0	0	0	0	0	1	0	0	0	1	1	Безусловный переход из 010 в 011
5	0	0	0	0	0	1	0	1	0	1	1	
6	0	0	0	0	0	1	1	0	1	0	1	Если $\bar{x}_1$, то 101 Иначе 100
7	0	0	0	0	0	1	1	1	1	0	0	

Окончание табл. 2

8	0	0	0	0	1	0	0	0	1	1	0	Безусловный переход из 100 в 110
9	0	0	0	0	1	0	0	1	1	1	0	
10	0	0	0	0	1	0	1	0	1	1	0	Безусловный переход из 101 в 110
11	0	0	0	0	1	0	1	1	1	1	0	
12	0	0	0	0	1	1	0	0	0	1	0	Если $\bar{x}_2$, то 010 иначе 111
13	0	0	0	0	1	1	0	1	1	1	1	
14	0	0	0	0	1	1	1	0	0	0	0	Безусловный переход из 111 в 000. Конец программы
15	0	0	0	0	1	1	1	1	0	0	0	

Допустим, исполняется программа с нулевым номером, т.е. регистр кода программы обнулен (рис. 1).

В исходном состоянии регистр кода состояния последующего и регистр сравнения обнулены сигналом системного сброса, неуказанного на рис. 1. Обнулены и регистр управления записью признаков и регистр PSW. Тогда адресные входы блока памяти автоматной модели также обнулены. Код последующего состояния не зависит от значения переменной, поэтому записан в двух строках, хотя на выходе мультиплексора в исходном состоянии – ноль.

Поэтому на выходе блока памяти установлен код следующего состояния 001_2. После начала программы с метки M_0 (табл. 1) микропроцессор выводит код следующего состояния 001_8 по адресу 200_8 с помощью логики записи информации в блок контроля в регистр сравнения записывается код 001_2, а в регистр кода состояния последующего – информация с выхода блока памяти автоматной модели. На выходе схемы сравнения так же, как и в исходном состоянии, поддерживается логическая единица.

Завершив выполнение линейного участка программы (табл. 1) микропроцессор выводит в регистр управления записью признаков код 300_8 = $= \underbrace{110}\,00\,\underbrace{000}_2$, что означает, что будет проверяться признак 110_2 (ноль – z), и это нулевая переменная x_0 (000). Далее выполняется подготовка к операции, устанавливающей значение признака. Пусть такой операцией будет вычитание из содержимого регистра – аккумулятора (A) содержимое регистра (B).

Перед выполнением собственно ветвления командой PUSH PSW выполняется загрузка регистра PSW. Если выполняется команда ветвления JNZ M1 с переменной $x_0 = 0$ (результат операции вычитания ненулевой), то производится переход на метку M1. Восстанавливается значение указателя стека командой POP PSW, далее опять выполняется установка регистра сравнения кодом 001_2.

Так как на выходе блока памяти автоматной модели в строке 2 установлен также 001_2 в случае, если действительно $x_0 = 0$, что зафиксировано на выходе мультиплексора и записано в триггер группы n триггеров логических условий, то при выполнении команд MVI 001, OUT 200 также произойдет сравнение информации.

В случае ошибки ветвления информация в регистре сравнения и в регистре кода состояния после выполнения соответствующих команд будет отличаться, что приведет к возникновению сигнала ошибки ветвления. Если выполняется команда ветвления JNZ M1 с переменной, $x_0 = 1$ (результат выполнения операции нулевой), то выполняется следующая команда POP PSW – восстанавливается значение указателя стека. На метке M6 фиксируется прохождение второго линейного участка (010_2) командами MVI 002_8, OUT 200_8. Так как на выходе блока памяти автоматной модели в строке 3 также 010_2 в случае, если действительно $x_0 = 1$, что зафиксировано на выходе мультиплексора и записано в триггер группы n триггеров логических условий, то при выполнении этой фиксации также произойдет сравнение информации. Иначе аналогично выше-

описанному будет зафиксирована ошибка ветвления.

Следует иметь в виду, что после команд записи в регистр управления записью признаков до выполнения команды PUSH PSW не должно быть команд, выполняющих запись в память во избежание неправильной записи в регистры признаков. Далее контроль ветвлений производится аналогично вышеописанному.

Безусловные переходы контролируются путем сравнения последующего состояния, записанного с выходов блока памяти автоматной модели в регистр кода состояния с фактически исполняемым кодом линейного участка в регистре сравнения. Блок памяти автоматной модели может быть реализован и на базе оперативной памяти.

При работе с некоторой другой программой в регистр кода программы выводится ее код и из блока памяти автоматной модели считывается соответствующая этой программе информация (в табл. 2 указана программа с нулевым номером – разряды адреса a_7, a_6).

Для контроля результатов операции в произвольной точке программы после таковой операции необходимо выполнить команды MVI с непосредственным операндом, устанавливающим номер проверяемого признака и номер дополнительной переменной; команду OUT 300, и команду фиксации прохождения соответствующего линейного участка.

Заключение

Предложенный метод может быть расширен на МПС с другими типами микропроцессоров.

Предлагаемый контроль хода выполнения программы позволяет повысить достоверность функционирования микропроцессорных систем.

Литература

1. Тюрин С.Ф., Харченко В.С., Тимонькин Г.Н., Мельников В.А. Программно-аппаратная реализация логических алгоритмов в микропроцессорных системах // Зарубежная радиоэлектроника. – 1992. – № 2. – С. 24-36.

2. Тюрин С.Ф., Тимонькин Г.Н., Харченко В.С. Методы аппаратной поддержки логических алгоритмов в микропроцессорных системах // Управляющие системы и машины. – 1993. – № 1. – С.55-63.

3. Тюрин С.Ф. Функционально-полные толерантные булевы функции // Наука и технология в России. – № 4. – 1998. – С. 7-10.

4. Тюрин С.Ф. Синтез адаптируемой к отказам цифровой аппаратуры с резервированием базисных функций // Приборостроение. – 1999. – № 1. – С. 36-39.

5. Тюрин С.Ф. Адаптация к отказам одновыходных схем на генераторах функций с функционально-полными толерантными элементами // Приборостроение. – 1999.– № 7. – С. 32-34.

6. Тюрин С.Ф. Проблема сохранения функциональной полноты булевых функций при «отказах» аргументов // Автоматика и телемеханика. – 1999. – № 9. – С. 176-186.

7. Тюрин С.Ф., Несмелов В.А., Харитонов В.А. и другие. Программируемое логическое устройство. Патент РФ № 2146840. Опубл. БИ № 8, 2000.

8. Тюрин С.Ф. и другие. Система для программного управления технологическим оборудованием. Патент РФ №2189623. Опубл. БИ № 26, 2002.

Поступила в редакцию 17.02.2006

Рецензент: д-р техн. наук, проф. В.С. Харченко, Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Харьков.