

## ПРИМЕНЕНИЕ ПРЯМОГО МОДЕЛИРОВАНИЯ В ДИАГНОСТИЧЕСКИХ ПРОЦЕДУРАХ

А.К. Олефир  
(Красноярск)

1°. В настоящее время известен ряд работ (см., например, [1 - 3]), в которых контрольные и диагностические программы составляются с помощью моделей анализируемых устройств. Считается, что для каждого типа элементов устройства известен список вариантов функционирования (в исправном и неисправном состояниях). Модель устройства составляется на таком уровне детализации, который обеспечивает зависимость выходных сигналов устройства от каждого варианта функционирования (в исправном и неисправном состояниях) каждого элемента, хотя бы на одном наборе входных переменных. Затем производится исследование на этих моделях каждого варианта функционирования каждого элемента для некоторого количества входных наборов и составляется таблица неисправностей, по которой осуществляется поиск отказавшего элемента.

Основной недостаток этого метода заключается в следующем: если  $l$  - количество элементов устройства,  $n$  - количество входных наборов, то возникает необходимость исследовать  $ln$  вариантов даже в том случае, если каждый элемент имеет только один тип отказа. Кроме того, этот метод удобно использовать только для поиска устойчивых отказов.

2°. В данной работе рассматривается один из возможных методов диагностики неисправностей, который состоит в следующем. Для тех операндов, при обработке которых в контролируемом устройстве получен неверный результат, решается модель с имитацией всех возможных неисправностей. Фиксируются те неисправности, при которых модель выдает результат, одинаковый с неверным результатом, полученным в контролируемом устройстве. В итоге получается подмножество неисправностей, которое содержит неисправность, вызвавшую данный отказ. Существенным преимуществом метода является возможность его применения к диагностике одиночных и программно-зависимых сбоев.

3°. Пусть имеется устройство, состоящее из элементов  $a_1, a_2, \dots, a_1, \dots, a_1$ . Каждый из элементов реализует некоторые функции  $f_1(a_1), f_2(a_1), \dots, f_k(a_1)$  в неисправных состояниях. Совокупность неисправностей

$$f_1(a_1), f_2(a_1), \dots, f_k(a_1), f_1(a_2), f_2(a_2), \dots, f_{k_2}(a_2), \dots, f_1(a_1), f_2(a_1), \dots, f_{k_1}(a_1)$$

образует множество  $F$ , которое по отношению к множеству входных наборов  $M = \{m_1, m_2, \dots, m_n\}$  обладает следующим свойством: если на наборе  $m_i$  проверяются неисправности

$$\Gamma_{m_i} = \{f_1^i, f_2^i, \dots, f_p^i\},$$

а на наборе  $m_j$  проверяются неисправности

$$\Gamma_{m_j} = \{f_1^j, f_2^j, \dots, f_q^j\},$$

то множества  $\Gamma_{m_i} \subset F$  и  $\Gamma_{m_j} \subset F$  ( $i \neq j; i, j = 1, 2, \dots, n$ ) образуют как пересекающиеся, так и непересекающиеся множества

ва  $[4]$ . Если  $\Gamma_{f_p}^{-1} \cap \Gamma_{f_q}^{-1} = M^*$ , где  $M^* \subset M$  и  $[4]|M^*| \geq 1$ , то неисправности  $f_p$  и  $f_q$  будем называть совместимыми.

Пусть в результате проведения тестовых процедур неверные результаты получены на наборах  $m_{i_1}, m_{i_2}, \dots, m_{i_k}$ .

Если  $\Gamma_{m_{i_1}} \cap \Gamma_{m_{i_2}} \cap \dots \cap \Gamma_{m_{i_k}} = F^*$ ,

где  $F^* \subset F$  и  $|F^*| > 1$ , то информация для локализации неисправного элемента является неполной. В этом случае не локализуется неисправный элемент. В настоящей работе при помощи прямого моделирования решается задача нахождения для заданного набора  $m_i$  полного множества совместимых неисправностей.

4<sup>0</sup>. Для работы модели по предложенному методу требуется следующая информация.

Г. Список неисправностей (СпиН). Для каждой неисправности  $f_i$  содержит слово следующего вида:

$p$	$t_\alpha$	$t_\beta$	$a_i$
-----	------------	-----------	-------

где  $p$  - адрес подпрограммы, описывающей неверное функционирование элемента;

$t_\alpha$ ,  $t_\beta$  - начальный и конечный такт работы со сбоем элемента, информация о которых хранится по адресу  $a_i$ .

2. Исходные данные и неверный результат в виде временной диаграммы анализируемого устройства.

Организующая программа, работающая совместно с моделирующей программой, заменяет элемент  $E$  неисправным элементом по списку неисправностей на модельное время в интервале  $t_\alpha - t_\beta$ . Для каждой неисправности  $f_i$  результат, полученный на модели, сравнивается с неверным результатом. В случае их совпадения неисправность  $f_i$  выдается на печать.

5<sup>0</sup>. Известно ( $[1]-[3]$ ,  $[5]-[12]$ ), что моделирование ЭВМ в устройстве ЭВМ ведется на трех уровнях: структурном, функциональном и логическом.

Программное воспроизведение работы устройства при наличии неисправностей предъявляет к модели специфические требования:

а) Имитация неисправности любого элемента, в том числе и вспомогательного, возможна на модели, которая учитывает соединения между элементами.

б) Неисправность в синхронной схеме может привести к тому, что эта схема превратится в асинхронную. Поэтому временной аспект в модели должен быть общим для синхронных и асинхронных устройств.

в) Неисправность элемента может выразиться в искажении фронта при переходе элемента из одного состояния в другое. Программное представление каждого типа элемента должно допускать эти (временные) изменения.

г) Должна быть обеспечена возможность просмотра состояния каждого элемента в любой дискретный момент времени.

Для решения поставленной задачи была использована система логического моделирования на уровне поэлементной детализации с учетом времени переходных процессов. Составными частями си-

стемы моделирования являются интерпретирующая программа (ИП), набор подпрограмм (по типам элементов) и модель схемы, которая описывается таблицей элементов и связей (ТЭС). Так как в нашем случае интерпретирующая программа в последовательные дискретные моменты времени вычисляет состояние каждого элемента схемы, просматривая прохождение сигналов по цепям, такое моделирование названо прямым (по Н.Я.Матихину). Опишем составные части системы моделирования.

Описание схемы. Информация об элементах и соединениях представляется в виде списка слов таблицы (ТЭС), в которой 1-му входу элемента  $E$  соответствует слово  $E_1$ :

S	p	$\tau$	t	$a_i$
---	---	--------	---	-------

$$i = 1, 2, \dots, l_{\text{вх}},$$

где  $l_{\text{вх}}$  - количество входов элементов схемы (длина ТЭС);  
 $S$  - сигнал на выходе элемента;  
 $p$  - тип элемента (адрес подпрограммы);  
 $\tau$  - признак переходного процесса ( $\tau = 0$  и  $\tau = 1$  - отсутствие и наличие переходного процесса, соответственно);  
 $t$  - содержимое счетчика переходного процесса данного элемента;  
 $a_i$  - адрес элемента, сигнал от которого поступает на  $i$ -ый вход элемента  $E$ .

Таким образом, каждое слово списка содержит информацию об элементе ( $S, p, \tau, t$ ) и об адресе связи с другим элементом ( $a_i$ ). Это позволяет наилучшим образом организовать память для работы интерпретирующей программы, так как каждый элемент связан со своим адресом в памяти. В качестве примера на рис.1 приведена ТЭС для схемы, изображенной слева.

Используемая символика имеет сходство с языком ЛИСП [12], однако здесь существенным является не порядок следования элементов списка, а отражение связей между элементами.

Воздействия. Внутренние и внешние воздействия привязаны к оси модельного времени. Внутренние воздействия определяются переходными процессами в элементах, а внешние - сигналами, поступающими на входы схемы. Внешние сигналы образуют список временной диаграммы (ВД), каждое слово которого имеет следующий вид:

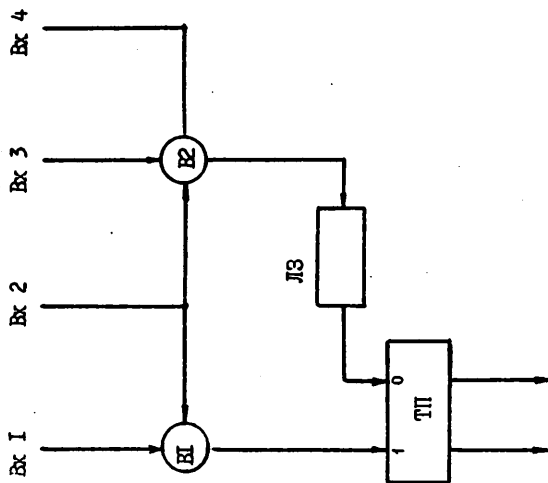


Рис. 1. Пример схемы.  
В<sub>1</sub>В<sub>2</sub> - вентили (схемы совпадения)  
ЛЗ - линия задержки  
ТП - потенциальный триггер.

Таблица элементов и соединений для приведенной схемы

Адреса ячеек памяти	S	P	τ	t	a <sub>j</sub>	Примечания
1000					1000	Вх. 1
1002					1002	Вх. 2
1004					1004	Вх. 3
1006					1006	Вх. 4
1010		7000			1000	Элемент В1
1012					1002	
1014		7000			1002	Элемент В2
1016					1004	
1020					1006	Линия задержки
1022		7500			1014	
1024		7500			1010	
1026					1022	

$s_{вх}$	$t_{вх}$	$a_{вх}$
----------	----------	----------

где:  $s_{вх}$  - значение сигнала, изменение которого произошло в момент  $t_{вх}$  ;  
 $a_{вх}$  - номер входа схемы, на котором происходит изменение сигнала.

Блок-схема интерпретирующей программы показана на рис.2 . ИП является универсальной программой, не требующей никаких изменений при переходе от одной схемы к другой. Каждому типу элементов соответствует отдельная подпрограмма. Отметим, что программное представление работы цифрового элемента основано на его свойстве воспринимать дискретную информацию. Изменение состояния элемента в модели производится с задержкой, соответствующей времени установления на входе элемента фиксированного уровня сигнала, необходимого для срабатывания данного элемента.

6°. Описываемый метод диагностики неисправностей реализуется системой программы, общее взаимодействие которых осуществляется организующей программой, находящейся в оперативной памяти (рис.4). Кроме описанных выше ИП и подпрограмм, в системе следует выделить программу "Корректор", которая расставляет сигналы временной диаграммы в соответствии с операндами, и программу "Контролер", которая сравнивает полученный на модели результат с неверным результатом проверяемого устройства.

7°. Метод реализован на ЭВМ "Урал-4". В оперативной памяти размещены ИП, подпрограммы элементов, временная диаграмма, ТЭС. Эталонная схема (без сигналов) находится на ИБ. На ИБ находятся СпИН, корректор, контролер.

Экспериментальная проверка метода проведена для диагностики неисправностей сумматора числа ЭВМ "М-20", который состоит из 560 цифровых элементов (36 разрядов числа, один дополнительный и 3 знаковых разряда). Модельное время разделено на 40 тактов. ЭВМ "Урал-4" решает эту модель за 2 минуты. В качестве неисправностей были определены типы функционирования:

- а) пропадание сигнала "1" в том случае, когда он должен быть на выходе элемента;
- б) появление сигнала "1" в том случае, когда на выходе должен быть "0" (помеха).



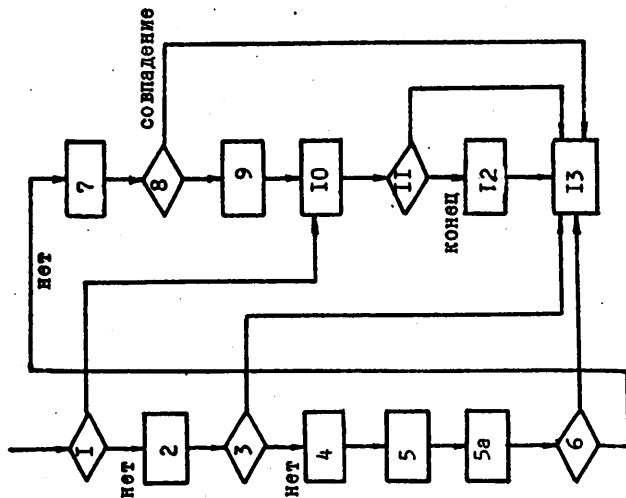


Рис.3. Блок-схема подпрограммы элемента совпадения на два входа (ламповый вентиль).

- 1 - проверка элемента  $E_i$  на переходный процесс;
- 2 - выделение  $a_{j1}$ ;
- 3 - проверка элемента по адресу  $a_{j1}$  на переходный процесс;
- 4 - запоминание выходного сигнала элемента  $a_{i1}$  (т.е.  $S_{\text{вх}1}$  - сигнала на первом входе элемента  $E_i$ );
- 5 - +  $t_1$  Сч ТЭС;
- 5a - выделение  $a_{j2}$ ;
- 6 - проверка элемента  $a_{j2}$  на переходный процесс;
- 7 - вычисление  $S^* = S_{\text{вх}1} \wedge S_{\text{вх}2}$   
(  $S_{\text{вх}2}$  - сигнал на втором входе элемента  $E_i$  )
- 8 - сравнение  $S^*$  и  $S$ , где  $S$  - выходной сигнал элемента  $E_i$  в момент времени  $t - t_1$ ;
- 9 - присвоение элементу  $E_i$  нового состояния  $S^*$ ;  
присвоение  $t = t_1$ ;
- 10 - + "1" счетчика переходного процесса;
- 11 - проверка окончания переходного процесса;
- 12 - снятие признака переходного процесса;
- 13 - выход в ИЛ.



- 1 - запись на МБ: корректор, контролер, СпИИ
- 2 - ввод в ОЗУ: ИП, подпрограмм, ТЭС;
- 3 - установка в "I" Сч СпИИ.
- 4 - ввод в ОЗУ перфокарт сбоя.
- 5 - останов ЭВМ при отсутствии перфокарт;
- 6 - вызов с МБ корректора и работа его;
- 7 - обновление ТЭС;
- 8 - вызов строки неисправности с МБ по Сч СпИИ;
- 9 - проба интервала времени;
- 10 - вызов контролера с МБ;
- 11 - работа ИП совместно с контролером;
- 12 - + "I" Сч СпИИ;
- 13 - проверка окончания СпИИ.

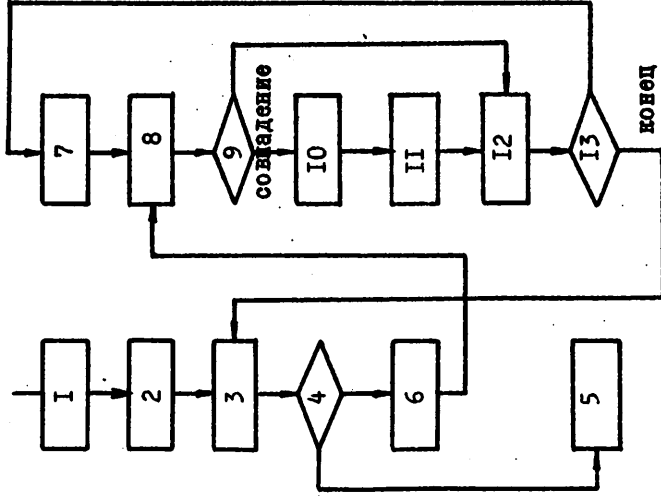


Рис. 4. Блок-схема взаимодействия программы

Эти неисправности представлены подпрограммами, адреса которых зафиксированы в строках СпИН.

В ЭВМ "М-20" выполняется программа, с помощью которой создается чередование режимов работы сумматора - легкого и тяжелого. Легкий режим создается за счет выполнения операции пересылки, однако никакие коды не пересылаются, поэтому сумматор практически не нагружается. Тяжелый режим создается при многократном (1500 раз) выполнении операции сложения. При этом в некоторых лампах происходят явления двух видов. Первое состоит в том, что, если лампа некоторое время заперта, эмиссионный слой катода "отравляется", и после отпираания лампа не сразу восстанавливает свои характеристики. Второе явление заключается в том, что в эмиссионный слой катода активное вещество поступает из глубинных слоев замедленно. При некоторой критической нагрузке эмиссионный слой не успевает восстанавливаться и характеристики лампы ухудшаются. Производится проверка полученного результата с эталонным. При несовпадении их выдается информация о сбое - операнды и неверный результат. Перфокарты с этой информацией вводятся вместе с описанной выше системой программ в ЭВМ "Урал-4" для получения списка совместимых неисправностей.

Метод опробован для операции сложения. Особый интерес представляет применение метода для локализации неисправностей, которые проявляются при выполнении многотактных операций (деление, умножение). В связи с тем, что время получения списка совместимых неисправностей зависит от величины СпИН, значительный интерес представляет получение оптимального списка для поиска неисправностей с заданной вероятностью.

Описанный метод ориентирован, в основном, на применение в вычислительных системах, так как в ВС всегда найдется исправная ЭВМ в качестве партнера к неисправному устройству или машине.

## Л И Т Е Р А Т У Р А

1. Диагностика неисправностей вычислительных машин. Сборник статей под редакцией Н.В. Паутина. Москва, "Наука", 1965.
2. Р.С. Гольдман, О.Ф. Немолочнов, В.П. Чипулис. Автоматизация построения тестов для проверки логических схем. "Тезисы докладов к Всесоюзному colloквиуму по автоматизации синтеза дискретных вычислительных устройств". Новосибирск, 1966.

3. А.К. Олсфир, А.И. Гракин, Г.А.Сенькина. О возможности программного поиска неисправностей в ЭЦВМ. "Материалы конференции молодых ученых СО АН СССР". Новосибирск, СО АН СССР, 1962.
4. К.Берж. Теория графов и её применения. Москва, ИЛ, 1962.
5. А.А. Уткин. О моделировании дискретных автоматов на ЭЦВМ. Сб. трудов "Вычислительные системы", выпуск 25, изд. "Наука", Сибирское отделение, Новосибирск, 1966.
6. И.Я. Ландау. Об автоматизации проектирования ЭЦВМ. "Автоматика и телемеханика", 1964, 25, № 11.
7. A.Baker. Computer simulation of digital systems. *Electro-Technol.*, 1964, 74, N3.
8. M.A.Bröner. Techniques for the simulation of computer logic. *Communications of the ACM*, 1964, vol.7, July.
9. R.P.Larsen, M.M.Mano. Modeling and Simulation of digital networks. *Communs.Assoc.Comput. Mach.*, 1965, 8, N5.
10. M.Lehman, R.Eshed, Z.Netter. The checking of computer logic by simulation on a computer. *Computer J.*, 1963, 6, N2.
11. G.Stockwell. Computer logic testing by simulation. *IRE Trans.Mill.Electron.*, 1962 Mil-6, N3.
12. McCarthy. Recursive functions of symbolic expressions and their computations by machine. *P.1. Communs of the ACM*, 1963, March, N 3.