

## УМЕНЬШЕНИЕ ВРЕМЕНИ МОДЕЛИРОВАНИЯ НЕИСПРАВНОСТЕЙ ЭВМ

*И.Б. Михайлов*

(Москва)

В последнее время широкое распространение в диагностике неисправностей ЭВМ получил метод диагностического словаря. Вкратце напомним основную идею данного метода. Составляется набор тестовых примеров. Полученный набор "проигрывается" с помощью программной модели исследуемой ЭВМ на рабочей машине (обычно универсальной ЭВМ с большим быстродействием и большим объемом памяти), с учетом внесенной в модель неисправности. После каждого тестового примера идет фиксация некоторой информации. Это может быть либо отметкой того, что правильно или неправильно выполнен пример, либо сохранение в памяти рабочей ЭВМ показаний некоторых регистров - моделей исследуемой ЭВМ. Накопленная информация обрабатывается по специальному алгоритму, в результате применения которого получаем двоичное слово - индекс соответствующей неисправности.

Проигрывая все неисправности заданного класса, получаем диагностический словарь, где против каждого индекса стоит наименование соответствующей ему неисправности. Очевидно, что если всего рассматривается  $N$  неисправностей, то необходимо иметь в индексе не менее  $\lg_2 N$  разрядов. Для выполнения однозначного соответствия между неисправностью и индексом необходимо брать большее количество разрядов. Однозначность в

сильной степени зависит от алгоритма вычисления индекса. Если применить взвешивание накопленной информации со случайными коэффициентами и с использованием кроме умножения и сложения еще и циклического сдвига вправо - влево, то, беря  $1,5 \lg_2 N$  разрядов в индексе, можно с большой достоверностью считать выполненной однозначность. Достоинством диагностического словаря является то, что здесь не требуется кропотливая и трудоёмкая работа по выявлению тестовых комбинаций, характерных для данной ЭВМ. Обычно начальный набор задается без труда разработчиком ЭВМ с учетом использования всех операций системы команд и всех основных режимов работы моделируемой ЭВМ. После первого "проигрывания", по спецпрограмме оценивается степень однозначности, выявляются множества неисправностей, имеющие один и тот же индекс, логически эквивалентные неисправности (операцию распознавания эквивалентности большей частью тоже можно автоматизировать) изымаются из вышеуказанных множеств и дальше пытаются изменить первоначальный набор или дополнить его новыми примерами и проводят вторичное "проигрывание". Если первоначальный набор только дополнялся новыми примерами, то моделируются только последние и в этом случае необходимо стыковать индексы первого и второго моделирования. Доопределение набора значительно легче в силу того, что основная масса неисправностей уже отсеивается при первом моделировании. В работе [1] отмечается, что подобные итерации над набором тестовых примеров довольно быстро приводит к желаемому результату.

В последнее время появился ряд работ [2],[3],[4], где рассматривается вопрос об автоматизации поиска тестовых комбинаций, но предлагаемые там методы либо рассматривают определенный тип схем и определенный класс неисправностей (комбинационные схемы, неисправности типа - постоянный "0", постоянная "1"), либо помогают найти тест, обнаруживающий неисправность и не обязательно ее локализирующий, либо описываются алгоритмы с чрезмерно большим объемом вычислений. По нашему мнению, в настоящее время нет реальной перспективы освобождения разработчика от подбора тестовых примеров. Однако нужно только облегчить эту работу.

К недостаткам диагностического словаря можно отнести слабую чувствительность к сбоям и неустойчивой работе схем. Вторым существенным недостатком - большой объем. Для современных ЭВМ множество неисправностей может достигать десятков тысяч

и, конечно, хранить такой словарь в виде печатного издания (одновременно и работать с ним) довольно затруднительно. Реальный выход из такой ситуации очевидно будет в применении магнитных лент. Неудобна также и корректировка словаря при изменении схемы ЭВМ, так как необходимо пересчитывать индексы. Однако намечаются пути устранения вышеперечисленных недостатков.

Составление диагностического словаря предъявляет серьезные требования к рабочей ЭВМ как в части быстродействия, так и в части объема памяти. Работа по моделированию начинается с составления вручную входного описания логики моделируемой ЭВМ. Специальный транслятор переводит входное описание в вид, удобный для рабочей ЭВМ (машинное описание). Входная кодировка должна обеспечивать детальное раскрытие схемы с точностью до элементов типа "И", "ИЛИ", "НЕ" и в то же время быть удобной для ручной работы. Если, например, моделируемая ЭВМ построена на основе набора стандартных ячеек, каждая из которых может содержать несколько схем "И", "ИЛИ", "НЕ", то целесообразно во входном описании ЭВМ дать список межъячеечных цепей (аналог монтажных таблиц) и описания каждой стандартной ячейки.

В силу того, что моделируется работа всей ЭВМ по программе, необходимо организовать работу всех блоков ЭВМ. Если мы опишем все блоки на уровне схем "И", "ИЛИ", "НЕ", то потребуется объем оперативной памяти, (для сокращения времени моделирования машинное описание желательно хранить в оперативной памяти) заведомо превышающий возможности существующих машин. Такое детальное описание всей ЭВМ в значительной степени увеличит и время моделирования. Поэтому возникает необходимость в использовании различных уровней кодировки логических схем.

Если, например, проводится в данное время моделирование неисправностей блока центрального управления, то нет необходимости блок арифметических операций подробно описывать; его работу можно моделировать по специальной подпрограмме. Таким же образом можно представить и блок памяти. Достоинство таких подпрограмм в том, что работа соответствующих блоков быстро моделируется и не требует большого объема оперативной памяти. Недостаток - необходимость составления и хранения подпрограмм, при этом можно применять некоторые стандартные приемы. Для уменьшения времени моделирования и объема памяти казалось бы нужно в оперативной памяти иметь детальные описания

только небольших олоков, неисправности которых последовательно рассматриваются, а остальные блоки представлять в виде подпрограмм. Однако, когда возникнет необходимость в переходе от олока к блоку, то потребуется время на замену подпрограмм подробной моделью. Вторая особенность заключается в том, что если "малых блоков" будет много, то придется затратить немалые усилия на программирование их индивидуальных подпрограмм. В то же время нетрудно, например, составить модель всего блока арифметики и сложнее моделировать малые блоки, так как возникают дополнительные проблемы сопряжения. Поэтому более целесообразно иметь небольшое количество блоков - подпрограмм (память, арифметика, блок контроля, блок ввода-вывода) и один крупный блок (например, центральное управление), описанный детально. Для сокращения времени работы последнего можно применить предлагаемый ниже метод.

Пусть, например, центральное управление (имеющее обычно несколько сотен элементов) описано с точностью до схем "И", "ИЛИ" и "НЕ". Разобьем блок на субблоки  $\sigma_{ij}$  (см. рисунок). Если мо-

C		
$\sigma_{11}$	$\sigma_{12}$	$\sigma_{13}$
$\sigma_{21}$	$\sigma_{22}$	$\sigma_{23}$
$\sigma_{31}$	$\sigma_{32}$	$\sigma_{33}$
D		

делируемая неисправность находится в центре субблока  $\sigma_{11}$ , то последний моделируется всегда поэлементно. Остальные субблоки используют свои списки типовых реакций. Каждый элемент списка содержит следующую информацию:

1. Набор входных сигналов (строка входов).
2. Набор выходных сигналов (строка выходов), как реакции на строку входов.
3. Время реакции.

Если субблок имеет  $n$  блоков, то всего возможно задать на вход  $2^n$  комбинаций. Но при реальной работе субблока по программе большая часть этих комбинаций не появляется, т.е. имеется ограниченное множество используемых строк входов. Перед моделированием эти типовые реакции неизвестны. Во время "проигрывания" первой неисправности субблока  $\sigma_{11}$  можно определить некоторые элементы используемого множества строк входов. Для определения реакции субблок моделируется поэлементно и строка

с получившейся реакцией заносится в список. При моделировании следующих неисправностей определяются другие элементы и так далее до определения всего используемого множества. Если найденные строки используются до заполнения всего множества, то субблок поэлементно не моделируется. Объем памяти, отводимый под список типовых реакций, ограничен и, следовательно, возникает необходимость управления этим списком. Даже если памяти достаточно, тем не менее список надо ограничивать, чтобы не получилось такого положения, что время поиска в списке будет больше времени поэлементного моделирования. Отметим еще следующий момент: если данная входная строка не изменяет состояние выхода субблока, то такие строки по возможности надо помещать в начале списка, только в этом случае типовое моделирование будет эффективней поэлементного (т.к. если вход не меняет выхода, то это означает, что входные сигналы "глушатся" где-то на первых каскадах - явление довольно частое в логических схемах). С другой стороны, если список ограничен, то сохранять необходимо только наиболее часто встречающиеся входные строки, применяя методы организации страничной памяти (в сильно упрощенном варианте). Обязательно надо обеспечить оперативную связь программиста со списками типовых реакций таким образом, чтобы по приказу с пульта рабочей ЭВМ объемы списков отдельных субблоков можно было регулировать, либо вообще запрещать, последнее полезно в случае, когда программисту ясно видно, что данный субблок не может иметь ограниченного эффективного набора типовых реакций (например, параллельные схемы свертки по модулю 2 и 3, многовходовые дешифраторы). Если моделируется неисправность элемента, находящегося на внутренних границах субблоков (элементы А и В), то поэлементно моделируются все смежные субблоки. На рисунке для элемента А субблоки  $\sigma_{21}$   $\sigma_{22}$  для элемента В субблоки  $\sigma_{22}$   $\sigma_{23}$   $\sigma_{32}$   $\sigma_{33}$ . Сложнее обстоит дело с элементами на внешней границе блока (С и Д на рисунке). Здесь необходимо вмешательство программиста, если неисправности от элементов С и Д распространяются внутрь блока, то моделирование их можно включить в данный блок, если во вне, то необходимо отнести к тем блокам, на которые они влияют. На практике подобные пограничные неисправности часто приходится обсчитывать вручную.

Третьим словом элемента списка типовых реакций является время. В этом слове немного разрядов, которые дают в условных

единицах время появления реакции относительно входного воздействия. Если реакция на выходе появляется в существенно различные моменты времени, то надо выделять несколько групп выходов со своими временами. Когда типовая входная строка меняет выход субблока, то последний переводится в возбужденное состояние (при помощи какой-либо метки в описании) на время, указанное в списке типовых реакций (если времен несколько, то по наибольшему). По истечении времени выходы данного субблока возбуждают входы других, а с него самого снимается возбужденное состояние и он готов к принятию нового воздействия. Отметим, что если воздействие не меняет выходов субблока, то оно игнорируется и субблок не возбуждается.

Представляет интерес случай, когда очередное воздействие застает субблок в возбужденном состоянии. В общем случае необходимо вернуться к первоначальному воздействию и провести поэлементное моделирование обоих воздействий. На практике, когда субблок имеет постоянное и относительно малое, порядка нескольких условных единиц (обычно условная единица равна среднему времени задержки на элемент), время реакции, то допустимо игнорировать первоначальное воздействие и отрабатывать более позднее.

Субблоки могут иметь элементы памяти. Если субблок имеет триггеров, которые имеют или не имеют связи с другими субблоками, то вводится строка состояний триггеров (со своими временами реакций) и субблок можно моделировать как конечный автомат. Но это усложнит дело в случае громоздкой таблицы переходов данного конечного автомата. Практически все запоминающие элементы либо всегда моделируются поэлементно, либо выделяются в отдельные субблоки. Такие элементы памяти, как регистры хранения кодов, не имеют структуры конечного автомата (т.к. состояние их зависит только от установки и сброса) и могут включаться в обычные субблоки.

В данной работе не рассматривается метод стыковки поэлементного и субблочного моделирования в силу того, что они сильно зависят от характера поэлементного моделирования.

В заключение отметим тот случай, когда применение метода типовых реакций наиболее эффективно. Если имеется возможность применить поэтапную диагностику (например, в двухмашинной системе), локализуя на первом этапе без помощи диагностического словаря неисправность с точностью до субблока, то для даль-

нейшего уточнения места неисправности надо применять диагностические словари каждого субблока. Для таких словарей легче составить тестовые примеры и что более важно - это, если составляется словарь субблока  $\sigma_{11}$ , то тестовые примеры проверяют его в различных режимах, в то время как остальные субблоки несут вспомогательные функции, обеспечивая общую работоспособность блока. Обычно отсутствует большое разнообразие входных воздействий для этих субблоков.

## Л и т е р а т у р а

1. Диагностика неисправностей вычислительных машин. "Наука", 1965 г.
2. J.M.Galey, R.E.Norby, J.P.Roth. Techniques for the Diagnosis of Switching Circuit Failures. IEEE Transactions on Communication and Electronics, vol. 83, N 74, p. 509-514.
3. D.B.Armstrong. On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets. IEEE Transactions on Electronic Computer, vol.15, N 1, pp. 66-73.
4. J.P.Roth. Diagnosis of Automata Failures: A Calculus and a Method. IBM Journal of Research and Development, vol.10, N 4, p. 278-291.