ISSN 0136-3549
0134-3823

TALLINNA
POLÜTEHNILISE INSTITUUDI
TOIMETISED

626

ТРУДЫ ТАЛЛИНСКОГО
ПОЛИТЕХНИЧЕСКОГО
ИНСТИТУТА

TPI '86

МАШИННОЕ ПРОЕКТИРОВАНИЕ ЭЛЕКТРОННЫХ УСТРОЙСТВ И СИСТЕМ

TPI '86

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

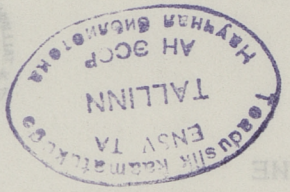
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 62-507+681.32+681.3+621.374.2

МАШИННОЕ ПРОЕКТИРОВАНИЕ ЭЛЕКТРОННЫХ УСТРОЙСТВ И СИСТЕМ

Электротехника и автоматика XXXI

Таллин 1986



ТАЛЛИНСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Труды ТПИ № 626

МАШИННОЕ ПРОЕКТИРОВАНИЕ ЭЛЕКТРОННЫХ УСТРОЙСТВ И СИСТЕМ

Электротехника и автоматика XXXI
На русском языке
Отв. ред. Э. Рюстерн
Техн. ред. В. Ранник
Сборник утвержден коллегией Трудов ТПИ 02.06.86
Подписано к печати 08.01.87. МВ-01306
Формат 60x90/16
Печ. л. 9,0 + 0,5
Уч.-изд. л. 8,74
Тираж 300
Зак. № 26
Цена 1 руб. 30 коп.
Таллинский политехнический институт,
200108 Таллин, Эхитаяте тее, 5
Ротапринт ТПИ, 200006 Таллин, ул. Коскла, 2/9

№ 626

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 62-507

П. Лейс

МЕТОДЫ ДЕКОМПОЗИЦИИ МИКРОПРОГРАММНЫХ АВТОМАТОВ

I. Введение

Алгебраическая теория декомпозиции конечных автоматов была разработана в работах Ю. Хартманиса и Р.Е. Стирнза [1]. Широкому практическому применению разработанной этими авторами теории препятствовала ориентация на малоэффективную табличную форму задания автомата. В данной работе методы декомпозиции применяются к модели микропрограммного автомата, позволяющего рассматривать сложные дискретные устройства управления.

Исследования, проводимые по декомпозиции микропрограммных автоматов, можно разделить на несколько направлений:

1) разработка методов (процедур) декомпозиции;

2) нахождение ортогонального множества декомпозиционных разбиений для получения сети с требуемыми свойствами;

3) разработка эффективных вычислительных алгоритмов для решения задач первых двух направлений.

В данной работе систематизируются результаты по разработке процедур декомпозиции микропрограммных автоматов.

2. Основные понятия

Микропрограммным автоматом называется система

$$M = (A, \{0,1\}^L, \{0,1\}^n, \delta, \lambda),$$

где A – множество внутренних состояний;

$\{0,1\}^L$ – входной алфавит;

$\{0,1\}^n$ – выходной алфавит;

$\delta: D(\delta) \to A$ – функция переходов, где $D(\delta) \subseteq A \times \{0,1\}^L$ – область определения функции δ;

$\lambda : D(\lambda) \to \{0,1\}^n$ – функция выходов, где $D(\lambda) \subseteq A$ – область определения функции λ.

МПА M называется полностью определенным, если $D(\delta) = A \times \{0,1\}^L$ и частичным, если $D(\delta) \subset A \times \{0,1\}^L$.

Наряду с понятием МПА будем использовать также понятие микропрограммного полуавтомата (МППА) $M_s = (A, \{0,1\}^L, \delta)$, не имеющего явно выделенных выходов.

Каждому переходу МПА из состояния a_j в состояние a_s ставим в соответствие комплекс 0 – кубов:

$$K^0(a_j, a_s) = \{ z \in \{0,1\}^L \mid \delta(a_j, z) = a_s \}.$$

Через $C(a_j, a_s)$ обозначен безызбыточный комплекс, покрывающий $K^0(a_j, a_s)$.

Отметим, что из-за большой мощности входного алфавита $\{0,1\}^L$ МПА чаще всего задаются множеством комплексов $\{C(a_j, a_s) \mid a_j, a_s \in A\}$, табличное представление которого называется таблицей переходов МПА. Таблица переходов МПА состоит из строк

$$r_i = (a_j, a_s, C(a_j, a_s)), \quad i = 1, \ldots, 4.$$

При решении задач декомпозиции в некоторых случаях удобнее пользоваться моделью МПА, в которой кроме двоичных входных каналов $x_1, \ldots, x_L$ имеется один абстрактный входной канал I. Таковым является модифицированный микропрограммный автомат (ММПА)

$$M_M = (A, \{0,1\}^L, I, \delta),$$

где A – множество состояний;
$\{0,1\}^L$ – структурный входной алфавит;
I – абстрактный входной алфавит;
$\delta : D(\delta) \to A$ – функция переходов, где $D(\delta) \subseteq A \times \{0,1\}^L \times I$.

В строки таблицы переходов добавляется также элемент $\theta \in I$:

$$r_i = (a_j, a_s, C(a_j, a_s), \theta).$$

Для постановки различных задач декомпозиции МПА необходимо понятие сети МПА.

Сетью микропрограммных автоматов (сетью МПА) называется система

$$N = (B, \{0,1\}^L, \{0,1\}^n, \Phi, g),$$

где 1°, $B = \{M_i \mid i \in J = \{1, \ldots, v\}\}$ – множество компонентных МПА (базис сети):

$$M_i = (A_i, \{0,1\}^{l_i+k_i}, \delta_i).$$

Здесь $\{0,1\}^{l_1}$ – внешний входной алфавит;

$\{0,1\}^{K_i}$ – внутренний входной алфавит (учитывает зависимость функционирования M_i от других компонентных автоматов сети);

$2^\circ\ \{0,1\}^{L}$ – входной алфавит сети;

$3^\circ\ \{0,1\}^{n}$ – выходной алфавит сети;

$4^\circ\ \Phi = \{\varphi_i \mid i \in J\}$ – множество функций соединения (структура сети),

$$\varphi_i : \underset{j \in J_i}{X} A_j \to \{0,1\}^{K_i},\ J_i \subseteq J;$$

$5^\circ\ g : \underset{j \in J}{X} A_j \to \{0,1\}^{n}$ – функция выходов сети.

Сети N можно ставить в соответствие МПА M_N, функционально эквивалентный N.

Результирующим МПА сети N называется микропрограммный автомат

$$M_N = (A_N, \{0,1\}^{L}, \{0,1\}^{n}, \delta_N, \lambda_N),$$

где $1^\circ\ A_N = \underset{i \in J}{X} A_i$;

$2^\circ\ \delta_N(a, z) = \underset{i \in J}{X} \delta_i(pr_i a, pr_{L_i} z, \varphi_i(pr_{J_i} a))$,

$z \in \{0,1\}^{L}$, $a \in A_N$, $L_i \subseteq \{1, \ldots, L\}$, $|L_i| = l_i$;

$3^\circ\ \lambda_N(a) = g(a)$, $a \in A_N$.

Реализацией микропрограммного автомата M_1 называется микропрограммный автомат M_2 (обозначается $R(M_1) = M_2$), если и только если у автомата M_2 существует изоморфный автомату M_1 подавтомат.

Декомпозицией микропрограммного автомата M называется сеть N, если и только если $R(M) = M_N$.

При разработке методов декомпозиции МПА используются различные операции с разбиениями и покрытиями. Приведем необходимые определения.

Покрытием φ на множестве A называется подмножество 2^A, удовлетворяющее условиям:

$$1^\circ\ \bigcup_{B \in \varphi} B = A;$$

$$2^\circ (\forall B \in \varphi)(\forall B' \in \varphi)\,[B \subseteq B' \Rightarrow B = B'].$$

Если дополнительно $B \cap B' = \Phi$ для всех $B \in \varphi, B' \in \varphi$, то φ называется разбиением на A.

Введем обозначения:

$$a \underset{\varphi}{\equiv} b \Leftrightarrow (\exists B \in \varphi)[a \in B \,\&\, b \in B];$$

$\varphi[a]$ – блок разбиения φ, содержащий элемент $a \in A$;

$\#(\varphi)$ – число блоков φ.

Произведением покрытий (разбиений) π и τ называется покрытие (разбиение)

$$\varphi = \pi \cdot \tau = \max(\{B' \cap B \mid B' \in \pi \,\&\, B \in \pi\}),$$

где оператор $\max$ удаляет из φ блоки, содержащиеся в других блоках φ.

Множество разбиений $P = \{\pi_i \mid i \in I\}$ называется ортогональным, если и только если

$$\prod_{i \in I} \pi_i = 0, \tag{I}$$

На множестве покрытий (разбиений) вводится отношение частичного порядка:

$$\tau \leq \pi \Leftrightarrow \tau \cdot \pi = \tau.$$

Пара разбиений (τ, π) на множестве состояний полностью определенного МПА M называется парой разбиений, если и только если

$$a_j \underset{\tau}{\equiv} a_k \Rightarrow \delta(a_j, z) \underset{\pi}{\equiv} \delta(a_k, z)$$

для всех $z \in \{0,1\}^L$; a_j; $a_k \in A$.

Через $M(\pi)$ обозначим максимальное разбиение, образующее пару разбиений с заданным разбиением π.

3. Общие методы декомпозиции МПА

Из общих методов декомпозиции рассматриваются общий метод декомпозиции полностью определенных МПА, общий метод декомпозиции частичных МПА и итеративный метод декомпозиции МПА.

3.1. Общий метод декомпозиции полностью определенных МПА [2]

Метод декомпозиции полностью определенного МПА $M = (A, \{0,1\}^L, \{0,1\}^n, \delta, \lambda)$ по ортогональному множеству разбиений P состоит в следующем:

I) Вычисляется множество разбиений $\{M(\pi_i) \mid i \in J\}$ по соотношению

$$a_j \underset{M(\pi_i)}{\equiv} a_k \Leftrightarrow (\forall B \in \pi_i)[C(a_j, B) = C(a_k, B)],$$

где

$$C(a, B) = \bigcup_{a_s \in B} C(a, a_s).$$

2) определяются множества индексов $\{J_i \mid i \in J\}$ таких, что

$$\prod_{j \in J_i} \pi_j \leq M(\pi_i), \quad J_i \subseteq J. \qquad (2)$$

Соединения компонентного автомата M_i с другими компонентными автоматами определяются множествами $J_i' = J_i \setminus \{i\}$. Поэтому для построения сети с минимальным числом связей между компонентными автоматами необходимо найти множество J_i минимальной мощности такое, что удовлетворяется (2).

3) Определяется множество разбиений $\{\theta_i \mid i \in J\}$:

$$\theta_i = \begin{cases} \prod_{j \in J_i'} \pi_j, & \text{если } J_i' \neq \Phi, \\ \text{единичное разбиение}, & \text{если } J_i' = \Phi. \end{cases}$$

Разбиение θ_i определяет внутренний абстрактный входной алфавит i-го компонентного ММПА: $I_i = \theta_i$.

4) Определяются компонентные ММПА (базис сети):

$$M_i = (A_i, \{0,1\}^{L_i}, I_i, \delta_i), \quad i \in J,$$

где

$$A_i = \pi_i;$$

$$\delta_i : \pi_i \times \{0,1\}^{L_i} \times I_i \longrightarrow \pi_i, \quad l_i \leq L.$$

Непосредственное вычисление функции $\delta_i, i \in J$ оказывается практически невозможным из-за большой мощности множества $\{0,1\}^L$. В [2] показано, что покрытие $C(B_\pi, B'_\pi \mid B_\theta)$ булевой функции, определяющей переход из состояния B_π в состояние B'_π при внутреннем входе $B_\theta \in I_i$, определяется следующим образом:

$$C(B_\pi, B'_\pi \mid B_\theta) \begin{cases} pr_{L_i} C(B_\pi \cap B_\theta, B'_\pi), & \text{если } B_\pi \cap B_\theta \neq \Phi; \\ \text{не определено}, & \text{если } B_\pi \cap B_\theta = \Phi, \end{cases}$$

где $L_i \subseteq \{1, \ldots, L\}, i \in J$ - множество существенных входных переменных автомата M_i.

5) Определяются функции соединения (структура сети)

$\varphi_i : \underset{j \in J_i}{X} A_j \longrightarrow I_i; \quad i \in J$ следующим образом:

$$\theta_i\Big[\bigcap_{j\in J_i'} pr_j a\Big], \text{ если } \bigcap_{j\in J_i'} pr_j a \neq \Phi;$$

не определено, если $\bigcap_{j\in J_i'} pr_j a = \Phi$,

где $a \in \underset{j\in J_i}{X} A_j$.

6) Определяется выходная функция сети:

$$g(a) = \lambda(a).$$

7) Закодируется внутренний входной алфавит компонентных ММПА. Для этого определяются взаимно однозначные функции

$$h_i : I_i \longrightarrow \{0,1\}^{k_i},\ i\in J;\ k_i = \lceil \log_2 I_i \rceil$$

здесь $\lceil t \rceil$ - наименьшее целое число, большее или равное. Обобщенные входные сигналы $C(B_{\pi_i}, B'_{\pi_i})$, $B_{\pi_i} \in A_i$, $B'_{\pi_i} \in A_i$, $i \in J$ определяются теперь следующим образом:

$$C(B_{\pi_i}, B'_{\pi_i}) = C\,(B_{\pi_i}, B'_{\pi_i} \mid B_{\theta_i}) \circ h(B_{\theta_i}),$$

где $B_{\theta_i} \in I$ и "$\circ$" - операция конкатенации кубов.

Тем самым сеть МПА, реализующая исходный микропрограммный автомат M, построена.

3.2. Общий метод декомпозиции частичного МПА [3]

Метод декомпозиции частичных МПА основывается на аппарате смешанных пар.

Пусть φ и π являются соответственно покрытием и разбиением на множестве состояний A.

Смешанной парой называется пара (φ, π), если и только если

$$a_j \underset{\varphi}{\equiv} a_s \Rightarrow \delta(a_j, z) \underset{\pi}{\equiv} \delta(a_s, z)$$

для всех $(a_j, z) \in D(\delta)$, $(a_s, z) \in D(\delta)$.

В [3] показано, что существует наибольшее покрытие $M_M(\pi)$, образующее смешанную пару с заданным разбиением π. Основное свойство покрытия $M_M(\pi)$ устанавливается следующей теоремой.

Теорема I. Пусть задано ортогональное множество разбиений $\{\pi_i \mid i \in J\}$ на множестве состояний A частичного МПА. Если множество разбиений $\{\pi_j \mid j \in J \subseteq J\}$ удовлетворяет условию

$$\bigcap_{j\in J_i} \pi_j \leqslant M_M(\pi_i), \qquad (3)$$

то функционирование компонентного автомата M_i не зависит от компонентных автоматов $\{M_j \mid j \in J \setminus J_i\}$.

Далее в [3] показано, что алгоритм декомпозиции частичных МПА отличается от алгоритма декомпозиции полностью определенных МПА лишь тем, что для нахождения множества индексов $\{J_i \setminus i \in J\}$, определяющих соединения в сети, вместо отношения (2) надо пользоваться отношением (3). Кроме того, для полностью определенных МПА справедливо $M_M(\pi) = M(\pi)$. Это означает, что предложенный метод декомпозиции частичных МПА является обобщением метода декомпозиции полностью определенных МПА.

Для временной сложности алгоритма декомпозиции МПА получена следующая оценка:

$$\tau_{МПА} \leq \Theta\left(\frac{m^2 t}{2} + 2H + m\right), \quad |J|,$$

где $m = |A|$ - число состояний декомпозируемого МПА,

$H = |\{c(a_j, a_k) \mid a_j, a_k \in A\}$ (длина структурной таблицы МПА).

В случае МПА средней сложности ($m = 50$, $H = 200$, $t = 10$, $L = 2^{30}$ 10) аналогичная оценка $\tau_{КА}$ алгоритма декомпозиции абстрактных автоматов превышает $\tau_{МПА}$ на несколько порядков.

3.3. Оптимизация базиса сети при декомпозиции МПА [4]

Во многих приложениях необходимо минимизировать длину таблицы переходов $H_i, i \in J$ компонентного МПА. Например, при реализации МПА на программируемых логических матрицах (ПЛМ) H_i должна удовлетворять условию $H_i \leq r$, где r - число термов ПЛМ.

Эта задача может быть решена целенаправленным выбором кодирующих функций h_i, $i \in J$ на шаге 7 алгоритма декомпозиции из 3.2.

Пусть заданы строки $r_i = (B_\pi^m, B_\pi^s, C(B_\pi^m, B_\pi^s), B_\theta^r)$ и $r_j = (B_\pi^t, B_\pi^u, C(B_\pi^t, B_\pi^n), B_\theta^v)$ таблицы переходов компонентного МПА, построенного по разбиению π, $B_\pi^m, B_\pi^s, B_\pi^t, B_\pi^u \in \pi$; $B_\theta^r, B_\theta^v \in \theta = I$.

Строки r_i и r_j называются сходными, $r_i \sim r_j$, если и только если

$$B_\pi^m = B_\pi^t,\ B_\pi^s = B_\pi^u,\ B_\theta^r \neq B_\theta^v \ \cup\ C(B_\pi^m, B_\pi^s) = C(B_\pi^t, B_\pi^u).$$

Если закодировать B_θ^r и B_θ^u соседними кодами (два кода называются соседними, если они отличаются лишь значением в одной координате), то строки r_i и r_j склеиваются в таблице переходов компонентного МПА.

Зададим теперь на алфавите I отношение $\psi_\sim$ следующим образом:

$$B_\theta^r \,\psi_\sim B_\theta^v \Leftrightarrow \exists r_i \exists r_j [r_i \sim r_j].$$

В [3] показано, что задача выбора кодирующей функции h, наиболее сокращающей длину H таблицы переходов компонентного МПА, сводится к задаче разбиения симметричного графа отношения $\psi_\sim$ на минимальное число полных подграфов.

Проведенные машинные эксперименты на потоке реальных и псевдослучайных МПА показали высокую эффективность предложенного метода кодирования внутреннего алфавита компонентного ММПА: во многих случаях длина перехода таблицы МПА сокращалась в 2-3 раза по сравнению с произвольно выбранной кодирующей функцией.

3.4. Итеративный метод декомпозиции МПА [5]

При декомпозиции МПА наибольшие вычислительные трудности возникают при определении ортогонального множества разбиений $\{\pi_i \mid i \in J\}$, по которому осуществляется декомпозиция. Известны алгоритмы для нахождения ортогонального множества разбиений, по которым можно построить параллельную, последовательную, параллельно-последовательную сеть или декомпозицию на сдвиговые регистры. Однако не существует метода для построения сети с произвольной заданной структурой. Кроме того, даже в случаях, когда в принципе существуют методы нахождения ортогонального множества разбиений с нужными свойствами, их практическое применение оказывается невозможным из-за огромных вычислительных трудностей. Для преодоления этих трудностей в [5] предлагается итеративный метод декомпозиции МПА. Итеративный метод декомпозиции МПА в определенном смысле позволяет объединить в единый процесс выбор разбиений и процесс декомпозиции.

Будем называть декомпозиционными неравенствами (ДНР) неравенства следующего вида:

$$\tau_j \geq \prod_{k \in K} \pi_k \cdot \prod_{l \in L} \delta_l, j \in J, \qquad (4)$$

где τ_j - известное покрытие или разбиение;
$\pi_k, k \in K$ - разбиения;
$\delta_l, l \in L$ - покрытия.

Некоторые разбиения π_k и покрытия δ_l известны, остальные требуется определить так, чтобы выполнялись неравенства (4).

Пусть задано простое ДНР

$$\tau \geq \pi_1 \cdot \pi_2,$$

где τ и π_1 известны, необходимо найти π_2.

Для этого введем понятие частной двух покрытий.

Частным покрытий (разбиений) α и β называется покрытие

$$\frac{\alpha}{\beta} = \sum \{\gamma \mid \gamma \cdot \beta = \alpha\},$$

которое можно вычислить следующим образом:

где
$$\frac{\alpha}{\beta} = A/R(\alpha, \beta),$$
$$R(\alpha, \beta) = \{(a, a') \quad a \underset{\alpha}{\equiv} a' \vee a \underset{\beta}{\equiv} a'\},$$

$A/R(\alpha, \beta)$ - фактор-множества отношения толерантности.

Итеративный метод декомпозиции МПА заключается в следующем.

Заданный МПА M декомпозируется на два автомата M_1 и M_2. Далее, в соответствии с выбранной структурой, декомпозируется автомат(ы) M_1 и/если M_2 и т.д.

Для выполнения этих разложений необходимо решать соответствующие ДНР.

4. Частные методы декомпозиции МПА

Несмотря на наличие общих методов декомпозиции МПА следует подчеркнуть важность разработанных специальных методов декомпозиции МПА, позволяющих получить сети МПА со специальными свойствами, что необходимо для успешного решения многих прикладных задач. Среди таких задач хочется особо выделить задачу синтеза контролепригодных устройств. Например, при реализации МПА параллельной сетью длина конт-

рольного эксперимента существенно сокращается по сравнению с реализациями, не использующими декомпозицию. Аналогичное утверждение справедливо и относительно реализации на сдвиговых регистрах.

4.I. Параллельная декомпозиция МПА [6]

Необходимым и достаточным условием для параллельной декомпозиции МПА является существование на множестве состояний ортогонального множества SP - разбиений.

SP - разбиением на множестве состояний A МПА называется разбиение π, удовлетворяющее условию

$$a_s \underset{\pi}{\equiv} a_t \;\&\; C(a_s, a_u) \cap C(a_t, a_v) \neq \Phi \Rightarrow a_u \underset{\pi}{\equiv} a_v . \qquad (6)$$

Алгоритм параллельной декомпозиции МПА существенно проще алгоритма общей декомпозиции МПА (в соответствии с шагами алгоритма 3.I):

I) $M(\pi_i) = \pi_i$, $i \in J$;

2) $J_i = \{i\}$, $i \in J$;

3) Θ_i = единичное разбиение;

4) $C(B_\pi, B'_\pi) = \bigcup\limits_{\substack{a_s \in B_\pi \\ a_m \in B'_\pi}} C(a_s, a_m)$, $B_\pi \in \pi$, $B'_\pi \in \pi$.

5) отсутствуют функции соединения.

4.2. Декомпозиция МПА на сдвиговые регистры [7]

При реализации МПА на базе сдвигового регистра длиной k выполняется:

$$\pi_{i-1} \leq M(\pi_i), \quad 2 \leq i \leq k, \qquad (7)$$
$$\#(\pi_i) = 2, \quad 1 \leq i \leq k,$$

отсюда непосредственно следует

$$\pi_{i-1} = M(\pi_i), \quad 2 \leq i \leq k.$$

Назовем (π, π') симметричной парой разбиений, если и только если между π и π' можно образовать взаимно однозначное соответствие

$$\psi : \pi \leftrightarrow \pi'$$

такое, что

$$\psi(B) = \pi'[\delta(a_m, z)],$$

где $B \in \pi$, $a_m \in B$ и $z \in \{0,1\}^l$.

Нетрудно показать, что пара разбиений (π_{i-1}, π) удовлетворяющая (7), является симметричной парой разбиений.

Таким образом, существование цепи $(\pi_1,\pi_2),(\pi_2,\pi_3),\ldots,(\pi_{k-1},\pi_k)$ симметричных пар разбиений (k-цепи) является необходимым и достаточным условием существований реализации МПА на базе сдвигового регистра длиной k.

Пусть (π_j, π'_j), $1 \le j \le l$, множество всех симметричных пар разбиений заданного МПА.

Из известных в алгебре пар соотношений следует, что тогда

$$\left(\prod_{j=1}^{l} \pi_j,\ \prod_{j=1}^{l} \pi'_j\right)$$

является также симметричной парой разбиений. При этом

$$\prod_{i=1}^{l} \pi_j = \pi^c \quad \text{и} \quad \prod_{j=1}^{l} \pi'_j = \pi^r$$

являются наименьшими разбиениями, которые образуют симметричную пару разбиений. Путем укрупнения блоков π^c и π^r можно получить всевозможные симметричные пары разбиений, в том числе образующие k-цепь.

В [6] предлагаются алгоритмы:

1) для нахождения $\pi^c = \{B_1^c, \ldots, B_\nu^c\}$ и

$\pi^r = \{B_1^r, \ldots, B_\nu^r\}$; $\psi(B_i^c) = B_i^r$, $1 \le i \le \nu$,

2) для образования k-цепей.

Теперь для реализации МПА на базе сдвиговых регистров следует выбрать минимальное число k-цепей таких, чтобы произведение разбиений из них было бы минимальным. Если это произведение нулевое, то можно применить общий алгоритм декомпозиции МПА, в противном случае необходимо дополнить множество разбиений до ортогонального.

4.3. Декомпозиция МПА на триггеры [8]

Одним из наиболее трудоемких этапов структурного синтеза МПА является минимизация системы функций возбуждения

элементов памяти. Эта задача упрощается, если заранее найти кратчайшие совокупности существенных переменных.

Пусть φ и π будут соответственно покрытием и разбиением на множестве состояний A микропрограммного автомата.

Пара (φ,π) называется смешанной J-парой, если и только если

1^0 π - двухблочное разбиение;

2^0 $(\varphi.\pi,\pi)$ является смешанной парой МПА.

Пара (φ,π) называется смешанной T-парой, если и только если

1^0 (φ,π) является смешанной J-парой;

2^0 $a_m \underset{\varphi}{\equiv} a_s \Rightarrow \delta(a_m,z) \not\underset{\pi}{\equiv} \delta(a_s,z)$

для всех (a_m,z), $(a_s,z) \in D(\delta)$.

Пара (φ,π) называется смешанной R-парой, если и только если

1^0 (φ,π) является смешанной J-парой;

2^0 $a_m \underset{\varphi}{\equiv} a_s \Rightarrow \delta(a_m,z) \underset{\pi}{\equiv} \delta(a_s,z) \vee$
$(\delta(a_m,z) \underset{\pi}{\equiv} a_m \,\&\, \delta(a_s,z) \underset{\pi}{\equiv} a_s)$

для всех (a_m,z), $(a_s,z) \in D(\delta)$.

В [8] было установлено существование наибольших покрытий $M_J(\pi)$, $M_T(\pi)$ и $M_R(\pi)$, образующих с разбиением π смешанную J-, T- и R-пару соответственно.

Пусть состояния из множества A будут закодированы с помощью кодирующей функции $q_\gamma: A \to \{0,1\}^m$. Тогда функция q_γ определяет множество двухблочных ортогональных разбиений $\{\pi_i \mid i \in J=\{1,\dots,\nu\}\}$ следующим образом:

$$a_m \underset{\pi_i}{\equiv} a_s \Leftrightarrow q_{\gamma i}(a_m) = q_{\gamma i}(a_s),$$

где $a_m, a_s \in A$, $q_{\gamma i} = pr_i q_\gamma$, $i \in J$.

В этом случае мы также говорим, что состояния из A закодированы в соответствии с ортогональным множеством разбиений $\{\pi_i \mid i \in J\}$. Далее в [8] доказана следующая теорема.

Теорема I. Совокупность внутренних переменных $\{q_k \mid k \in J \setminus J_i\}$ несущественна для функции возбуждения i-го JK-триггера, если и только если

$$\prod_{j \in J_i} \pi_j \le M_J(\pi_i), \quad J_i \subseteq J. \qquad (8)$$

Аналогичные результаты можно получить для T- и RS-триггеров, если в (8) $M_J(\pi_i)$ заменить на $M_T(\pi_i)$ и $M_R(\pi_i)$ соответственно.

Теперь очевидно, что для декомпозиции МПА на JK-триггеры (в сеть, в которой лишь некоторые компонентные автоматы являются триггерами), можно пользоваться общим методом декомпозиции МПА, если отношение (3) заменить отношением (8).

Конечно, не требуется и определить функцию перехода для JK-триггера. Аналогично модифицируется алгоритм декомпозиции и в случае использования других типов триггеров.

4.4. Линейная декомпозиция [5]

Сеть МПА называется линейной, если компонентные автоматы M_i, $i \in J$ можно упорядочить таким образом, что между собой связаны только автоматы M_i и M_{i+1}, $1 \le i \le v-1$.

Вполне очевидно, что для линейной сети из (3) следует:

$$\begin{aligned}
&M_M(\pi_1) \ge \pi_1 \cdot \pi_2 \\
&M_M(\pi_2) \ge \pi_1 \cdot \pi_2 \cdot \pi_3 \\
&\cdots\cdots\cdots \\
&M_M(\pi_i) \ge \pi_{i-1} \cdot \pi_i \cdot \pi_{i+1} \\
&\cdots\cdots\cdots \\
&M_M(\pi_v) \ge \pi_{v-1} \cdot \pi_v \\
&\prod_{j=1}^{v} \pi_j = 0.
\end{aligned}$$

Если π_1 известно, то система (9) решается следующим образом:

$$\pi_2 \le \frac{M_M(\pi_1)}{\pi_1}$$

откуда

$$\pi_3 \le \frac{M_M(\pi_2)}{\pi_2}$$

$$\cdots\cdots\cdots$$

$$\pi_{i+1} \le \frac{M_M(\pi_i)}{\pi_{i-1} \cdot \pi_i}$$

$$\cdots\cdots\cdots$$

$$\pi_N \le \frac{M_M(\pi_{v-1})}{\pi_{v-1}}.$$

Таким образом, рассматривая линейную декомпозицию как частный случай итеративной декомпозиции, несложно получить соотношения для последовательного нахождения декомпозиционных разбиений.

Литература

I. Hartmanis J., Stearns R.E. Algebraic structure theory of sequential machines. - Englewood Cliffs, N. Y., Prentice-Hall Inc., 1966, p. 211.

2. Jakobson G., Keevallik A., Leis P. Some aspects of the construction of microprogram automata networks. - In: IFAC-symposium "Discrete Septems", Dresden, 1977, p. 120-128.

3. Лейс П. Определение структуры сети при декомпозиции неполностью определенных микропрограммных автоматов. - Тр. Таллинск. политехн. ин-та, 1977, № 432, с. 57-68.

4. Лейс П., Эллервеэ П. Оптимизация базиса сети при декомпозиции микропрограммных автоматов. - Тр. Таллинск. политехн. ин-та, 1985, № 601, с. 19-25.

5. Лейс П., Салум К. Итеративный метод декомпозиции конечных автоматов. - Тр. Таллинск. политехн. ин-та, 1983, № 550, с. 81-89.

6. Лейс П., Якобсон Г. О параллельной декомпозиции микропрограммных автоматов. - Тр. Таллинск. политехн. ин-та, 1976, № 409, с. 129-137.

7. Лейс П., Кеэваллик А., Круус М. О реализации микропрограммных автоматов на сдвиговых регистрах. - Тр. Таллинск. политехн. ин-та, 1984, № 577, с. 85-95.

8. Лейс П. Применение смешанных пар для нахождения существенных переменных функций возбуждения. - В кн.: Многопроцессорные вычислительные структуры. Таганрог, 1979, вып. I (X), с. 49-51.

P. Leis

Microprogram Automata Decomposition Methods

Abstract

General and special methods for the decomposition of microprogram automata are presented in a systematic way. The following decomposition methods are included:

- general decomposition of completely and incompletely specified microprogram automata;
- iterative decomposition method;
- method for the parallel decomposition;
- shift-register decomposition;
- flip-flop decompositions;
- linear decomposition.

№ 626

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 62-507

А.В. Судницын, В.Р. Вийес,

ЗАДАЧА ВЫБОРА РАЗБИЕНИЙ В ДЕКОМПОЗИЦИОННОМ СИНТЕЗЕ ДИСКРЕТНЫХ УПРАВЛЯЮЩИХ АВТОМАТОВ

Введение. Использование программируемых матриц и микропроцессоров во встроенных системах управления, в частности, измерительными комплексами значительно расширяет функциональные возможности последних. При этом существенное значение приобретает проблема синтеза дискретных управляющих автоматов в базисе программируемых больших интегральных схем (БИС). Однако на отдельной БИС может быть реализован автомат, не превышающий по сложности некоторого ресурса (число внешних полюсов БИС, быстродействие программной реализации и т.д.). Поэтому мы вынуждены строить реализации в виде сетей из БИС. Одним из наиболее перспективных подходов, которые могут быть применены к решению сложной и трудоемкой задачи построения сетей из БИС, является развитие методов декомпозиции [1, 2, 3]. Этот подход характерен тем, что в результате декомпозиции автомат разлагается в сеть автоматов таким образом, что ее компонентные автоматы и вся сеть в целом удовлетворяла бы заданным требованиям. При решении практических задач декомпозиции целесообразно опираться на аппарат алгебры пар разбиений [4]. Решение задач декомпозиции упирается при этом в проблему нахождения соответствующих полных (ортогональных) систем разбиений на множествах состояний, входов и выходов автомата. Хорошо известно, что последняя задача относится к классу NP-полных задач и не может быть практически решена путем полного перебора вариантов. Настоящая работа посвящена разработке подхода к генерированию полных систем разбиений для получения декомпозиции с заданными свойствами. Для описания алгоритма функционирования управляющих автоматов, следуя работам [1, 5],

мы будем использовать модель микропрограммного автомата (МПА). Формально МПА определим как пятерку

$$A = (S, X, Y, \delta, \lambda),$$

где $S = \{s_i / i = 1,\ldots,M\}$ – множество внутренних состояний;

$X = \{x_i / i = 1,\ldots,L\}$ – множество входных двоичных переменных;

$Y = \{y_i / i = 1,\ldots,N\}$ – множество выходных двоичных переменных;

$\delta : S \times \{0,1\}^L \longrightarrow S$ – функция переходов;

$\lambda : S \longrightarrow \{0,1\}^N$ – функция выходов.

Модель МПА позволяет более компактную запись автоматных алгоритмов управления, благодаря использованию обобщенных входных сигналов $X(s_m, s_n)$ – конъюнкции входных переменных МПА А. Автомат переходит из состояния s_m в состояние s_n под действием всех тех входных наборов $z \in \{0,1\}^L$, на которых конъюнкция $X(s_m, s_n) = 1$.

Постановка задачи выбора системы разбиений. Сетевые реализации МПА на базе БИС (программируемые логические матрицы и микропроцессорные комплекты) налагают требования на параметры сети МПА, получаемой в результате декомпозиции. Так к программным реализациям выдвигают требования по быстродействию, а также при этом должны быть учтены ограничения со стороны объема памяти у базисного элемента.

Опираясь на анализ перечисленных выше свойств, проведенный в работах [2, 3], дадим формальный критерий (параметр), определяющий целесообразность применения декомпозиционных методов синтеза, приводящих к реализациям в виде сетей из БИС.

Назовем G-параметром МПА А максимальное число из множества

$$\{ |Z(s_m)| / s_m \in S \},$$

где $|Z(s_m)|$ – мощность множества входных двоичных переменных, существенных для вычисления перехода из состояния s_m.

Обозначим через $\mathcal{M}(G = q)$ – МПА с G-параметром, равным q. Через $\mathcal{P}(G \leq t)$ обозначим сеть МПА, компонентные автоматы которой имеют G-параметр не больше, чем t.

В настоящей работе рассмотрена задача декомпозиции МПА $\mathcal{M}(G=g)$ в сеть $\mathcal{F}(G \leqslant t)$.

Пусть $B \subseteq S$. Через $C(\delta, B)$ обозначим безызбыточное покрытие комплекса 0-кубов [I, 5]

$$K^0(\delta, B) = \{ z \in \{0,1\}^L / \delta(\delta, z) \in B \}.$$

Через $F_c(\delta, B)$ обозначим множество входных двоичных переменных, соответствующих компонентам, связанным во всех кубах из $C(\delta, B)$.

Нетрудно показать, что из процедуры построения сети автоматов [I] непосредственно следует, что решение рассматриваемой в настоящей работе задачи декомпозиции сводится к нахождению ортогонального множества разбиений $P=\{\pi_i / i=1, \ldots, n\}$, обладающего следующими свойствами

$$\forall \pi \in P \; \forall B \in \pi \; \forall \delta \in S \; [|F_c(\delta, B)| \leqslant t].$$

Аппарат неполностью определенных разбиений. Учитывая сложность решаемых задач переборного характера с целью сокращения перебора вариантов, используем понятие неполностью определенного разбиения.

Неполностью определенным разбиением (НОР) на множестве S называется совокупность подмножеств множества S таких, что $B_i \neq \phi (i=1, \ldots, m)$, $B_i \cap B_j = \phi (i \neq j)$, $\bigcup_{i=1}^{m} B_i = S \setminus \delta_d$

δ_d - специальный блок НОР.

НОР ρ можно интерпретировать как компактную запись целой совокупности $\Gamma(\rho)$ разбиений на S. Причем $\pi \in \Gamma(\rho)$, если и только если для каждого $B' \in \pi$ либо найдется блок $B \in \rho$ такой, что $B \subseteq B' \subseteq B \cup \delta_d$, либо $B' \subseteq \delta_d$. При преобразовании НОР ρ в $\pi \in \Gamma(\rho)$ элементами из δ_d могут быть дополнены неспециальные блоки из ρ или образовываться новые блоки разбиения π.

НОР ρ_1 не меньше НОР ρ_2 ($\rho_1 \geqslant \rho_2$), если и только если для каждого $\pi_1 \in \Gamma(\rho_1)$ найдется $\pi_2 \in \Gamma(\rho_2)$ такое, что $\pi_1 \geqslant \pi_2$.

Произведение НОР ρ_1 и ρ_2 называется НОР $\rho_1 \cdot \rho_2$, специальный блок которого образуется объединением специальных блоков НОР ρ_1 и ρ_2, а для неспециальных блоков справедливо, что элементы δ и δ' принадлежат одному и тому же блоку, ес-

ли и только если они принадлежат одному неспециальному блоку как в ρ_1, так и в ρ_2.

Суммой НОР ρ_1 и ρ_2 называется НОР $\rho_1+\rho_2$, специальный блок которого образуется пересечением специальных блоков НОР ρ_1 и ρ_2, а для его неспециальных блоков справедливо, что элементы δ и δ' принадлежат одному и тому же блоку, если и только если существует последовательность $\delta=\delta_0,\delta_1,\ldots,\delta_\kappa=\delta'$ такая, что для всех $i\in\{0,\ldots,\kappa-1\}$ δ_i и δ_{i+1} принадлежат одному неспециальному блоку в ρ_1 или в ρ_2.

Множество НОР называется полным, если произведение входящих в него НОР равно НОР, все блоки которого, кроме специального, состоят из одного элемента.

Нетрудно доказать [6], что множеству НОР $\mathcal{P}=\{\rho_i/i=1,\ldots,n\}$ можно поставить в соответствие полное множество разбиений $P=\{\pi_i\in\Gamma(\rho_i)/i=1,\ldots,n\}$, если и только если множество НОР полное.

В исходном МПА каждому состоянию δ и входной двоичной переменной x можно сопоставить минимальное НОР $\sigma(\delta,x)=(B_1,\ldots,B_m,b_d)$ такое, что для всех его неспециальных блоков $B_i, i=1,\ldots,m$ в комплексе кубов $C(\delta,B_i)$ координата, соответствующая x - свободная.

Первичным σ-разбиением для состояния $\delta_m\in S$ называется НОР $\sigma_m(\mathcal{K})=\sum_{\kappa\in\mathcal{K}}\sigma(\delta_m,\kappa)$,

где $\mathcal{K}\subseteq Z(\delta_m)$ и $|\mathcal{K}|=|Z(\delta_m)|-t$.

Из определения следует, что первичное σ-разбиение есть НОР на S такое, что для любого неспециального блока B из $\sigma_m(\mathcal{K})$ координаты в $C(\delta_m,B)$, соответствующие всем переменным из $\mathcal{K}$ - свободные.

Состоянию $\delta_m\in S$ в общем случае можно сопоставить несколько первичных σ-разбиений (но не более, чем $C^{|Z(\delta_m)|-t}_{|Z(\delta_m)|}$ - число сочетаний из $|Z(\delta_m)|$ по $|Z(\delta_m)|-t$). В случае, если $Z(\delta_m)\leqslant t$, будем считать, что множество первичных σ-разбиений для состояния δ_m включает лишь НОР, которое состоит только из одного специального блока.

σ-разбиением r-го порядка называется НОР $\sigma^r=\sum_{m=1}^{M}\rho_m$, где ρ_m - первичное σ-разбиение для состояния δ_m.

Метод выбора системы разбиений. Предлагаемый метод построения ортогонального множества разбиений сводится к построению множества НОР, из которого путем доопределения может быть получено искомое множество декомпозиционных разбиений P.

Лемма I. Максимальное число внешних входных переменных, существенных для вычисления следующего состояния компонентного автомата A_i, соответствующего декомпозиционному разбиению π_i из P, не превосходит некоторого заданного числа t ($t<g$), если и только если существует σ-разбиение r-го порядка σ^r такое, что $\pi_i \geqslant \sigma^r$ и $r = g - t$.

Доказательство. Нетрудно показать, что G-параметр компонентного автомата, соответствующего разбиению π_i, равен максимальному числу из множества $\{|F_c(s,B)| / s \in S,\ B \in \pi_i\}$.

Пусть σ^r есть σ-разбиения r-го порядка на множестве состояний декомпозируемого МПА. Образуем разбиение $\tau \in \Gamma(\sigma^r)$, блоки котсрого либо совпадают с неспециальными блоками из σ^r, либо состоят из одного элемента, принадлежащего специальному блоку НОР σ^r. Тогда по определению понятия σ-разбиения максимальное число из множества $\{|F_c(s,B)| / s \in S, B \in \tau\}$ совпадает с $t = g - r$. Очевидно, что для любого разбиения π', которое не меньше чем τ, ни одно число из множества $\{|F_c(s,B')| / s \in S, B \in \pi'\}$ не превосходит t.

Необходимость условия существования можно доказать от противного. Предположим, что не существует σ-разбиения r-го порядка σ^r такого, что $\sigma^r \leqslant \pi_i$. Тогда найдется такой блок $B \in \pi_i$, что $|F_c(s,B)| \geqslant t, s \in S$. Последнее противоречит тому, что G-параметр компонентного автомата, соответствующего разбиению π_i, не превосходит t. Следовательно, предположение неверно, и существование σ-разбиения r-го порядка σ^r такого, что $\sigma^r \leqslant \pi_i$, является необходимым условием.

Пусть $\mathcal{P}_\sigma^r$ - полное множество σ-разбиений r-го порядка. Справедлива следующая теорема.

Теорема. Существование $\mathcal{P}_\sigma^r$ для МПА $\mathcal{M}(G=g)$ необходимо и достаточно для его декомпозиции в сеть $\mathcal{P}(G \leqslant g - r)$.

Доказательство. Непосредственно следует из основной теоремы декомпозиции МПА парным методом [I] и леммы I.

Приведенная теорема служит обоснованием генерации полного (ортогонального) множества разбиений P как последовательности следующих этапов:

I) получение множества первичных σ-разбиений,

2) получение полного множества НОР r-го порядка,

3) преобразование полного множества НОР в полное множество разбиений.

Если в качестве оптимизационного критерия взять минимум компонентных автоматов сети, получаемой в результате декомпозиции, тогда на втором этапе получение полного множества НОР сопряжено с минимизацией его мощности. Комбинаторная сложность этой задачи обуславливает необходимость разработки приближенного алгоритма. Используемая при этом эвристика может состоять в том, что искомое (в начале пустое) множество на каждом шаге дополняется одним НОР. Причем в качестве очередного НОР выбирается такое НОР, произведение которого с найденными минимально.

В качестве примера возьмем МПА с прямой таблицей переходов, приведенной на рис. I. Максимальным является число входных переменных, существенных для вычисления состояния перехода из первого состояния, т.е. $g = 4$. Требуется построить декомпозицию для случая $t = 2$. Последняя задача сводится к нахождению полного множества σ-разбиений 2-го порядка $(g - t = 2)$.

Для первого состояния имеем

$$\sigma(1, x_1) = (\overline{1,2,3,5};\ \overline{4,6,7,8};\ \langle\ \rangle);$$

$$\sigma(1, x_2) = (\overline{1,2};\ \overline{7,8};\ \langle 3,4,5,6\rangle\ );$$

$$\sigma(1, x_3) = (\overline{3,5};\ \overline{4,6};\ \langle 1,2,7,8\rangle\ );$$

$$\sigma(1, x_4) = (\overline{1,7};\ \overline{2,8};\ \overline{3,4};\ \overline{5,6};\ \langle\ \rangle).$$

Первичными σ-разбиениями для первого состояния будут следующие НОР:

$$\sigma_1(x_1,x_2) = \sigma(1,x_1) + \sigma(1,x_2) = (\overline{1,2,3,5};\ \overline{4,6,7,8};\ \langle\ \rangle) = \sigma_1(x_1,x_3)$$

$$\sigma_1(x_2,x_3) = \sigma(1,x_2) + \sigma(1,x_3) = (\overline{1,2};\ \overline{7,8};\ \overline{3,4};\ \overline{5,6};\ \langle\ \rangle);$$

$$\sigma_1(x_2,x_4) = \sigma(1,x_2) + \sigma(1,x_4) = (\overline{1,2,7,8};\ \overline{3,4};\ \overline{5,6};\ \langle\ \rangle);$$

$$\sigma_1(x_3,x_4) = \sigma(1,x_3) + \sigma(1,x_4) = (\overline{1,7};\ \overline{2,8};\ \overline{3,4,5,6};\ \langle\ \rangle):$$

Исходное состояние	Сост. перех.	Входной сигнал
1	1	$x_1x_2x_4$
	2	$x_1\bar{x}_2x_4$
	3	$\bar{x}_1x_3x_4$
	4	$\bar{x}_1x_3x_4$
	5	$\bar{x}_1\bar{x}_3x_4$
	6	$\bar{x}_1\bar{x}_3\bar{x}_4$
	7	$x_1x_2\bar{x}_4$
	8	$x_1\bar{x}_2\bar{x}_4$
2	3	x_2x_4
	4	$x_2\bar{x}_4$
	5	$\bar{x}_2x_4$
	6	$\bar{x}_2\bar{x}_4$
3	3	x_3x_4
	5	$\bar{x}_3x_4$
	7	$\bar{x}_4x_5$
	8	$\bar{x}_4\bar{x}_5$
4	1	$\bar{x}_4x_5$
	2	$\bar{x}_4\bar{x}_5$
	3	x_3x_4
	5	$\bar{x}_3x_4$
5	5	x_4
	7	$\bar{x}_4x_5$
	8	$\bar{x}_4\bar{x}_5$
6	1	$\bar{x}_4x_5$
	2	$\bar{x}_4\bar{x}_5$
	5	x_4
7	1	x_1x_2
	2	$x_1\bar{x}_2$
	3	$\bar{x}_1x_3$
	5	$\bar{x}_1\bar{x}_3$
8	3	x_2
	5	$\bar{x}_2$

Рис. 1. Прямая таблица переходов декомпозируемого МПА.

Нетривиальное множество первичных σ-разбиений будет также у третьего, четвертого и седьмого состояний.

$\sigma_3(x_3) = \sigma(3,x_3) = (\overline{3,5};\ \langle 1,2,4,6,7,8\rangle\ ) = \sigma_4(x_3) = \sigma_7(x_5)$;

$\sigma_3(x_5) = \sigma(3,x_5) = (\overline{7,8};\ \langle 1,2,3,4,5,6\rangle\ ) = \sigma_7(x_5)$;

$\sigma_4(x_5) = \sigma(4,x_5) = (\overline{1,2};\ \langle 3,4,5,6,7,8\rangle\ ) = \sigma_7(x_2)$.

Множество σ-разбиений 2-го порядка состоит из пяти НОР

$\sigma_1^2 = (\overline{1,2,3,5};\ \overline{4,6,7,8};\ \langle\ \rangle)$; $\sigma_2^2 = (\overline{1,2};\ \overline{3,5};\ \overline{4,6};\ \overline{7,8};\ \langle\ \rangle)$;

$\sigma_3^2 = (\overline{1,2,7,8};\ \overline{3,4,5,6};\ \langle\ \rangle)$; $\sigma_4^2 = (\overline{1,2,7,8};\ \overline{3,4};\ \overline{5,6};\ \langle\ \rangle)$:

$\sigma_5^2 = (\overline{1,7};\ \overline{2,8};\ \overline{3,4,5,6};\ \langle\ \rangle)$.

Исход. сост.	Сост. перех.	Входной сигнал
1	b_1 b_2	x_4 $\bar{x}_4$
2	b_1 b_2	x_4 $\bar{x}_4$
3	b_1 b_2	x_4 $\bar{x}_4$
4	b_1	1
5	b_1 b_2	x_4 $\bar{x}_4$
6	b_1	1
7	b_1	1
8	b_1	1

Рис. 2. Таблица функции $F_1 : S \times \{0,1\}^L \to \pi_1$.

Полным множеством НОР минимальной мощности и декомпозиционными множествами будут следующие разбиения

$\pi_1 = (\overline{1,2,3,5} = b_1\ ;\ \overline{4,6,7,8} = b_2)$;

$\pi_2 = (\overline{1,2,7,8} = d_1 ;\ \overline{3,4} = d_2 ;\ \overline{5,6} = d_3);$

$\pi_3 = (\overline{1,7} = c_1 ;\ \overline{2,8} = c_2 ;\ \overline{3,4,5,6} = c_3).$

Функции $F_i : S \times \{0,1\}^L \to \pi_i$, по которым строится сеть автоматов [I], представлены на рис. 2, 3 и 4.

Исход. сост.	Сост. перех.	Входной сигнал
1	c_1 c_2 c_3	x_1x_2 $x_1\bar{x}_2$ $\bar{x}_1$
2	c_3	1
3	c_1 c_2 c_3	$\bar{x}_4x_5$ $\bar{x}_4\bar{x}_5$ x_4
4	c_1 c_2 c_3	$\bar{x}_4x_5$ $\bar{x}_4\bar{x}_5$ x_4
5	c_1 c_2 c_3	$\bar{x}_4x_5$ $\bar{x}_4\bar{x}_5$ x_4x
6	c_1 c_2 c_3	$\bar{x}_4x_5$ $\bar{x}_4\bar{x}_5$ x_4
7	c_1 c_2 c_3	x_1x_2 $x_1\bar{x}_2$ $\bar{x}_1$
8	c_3	1

Рис. 3. Таблица функции $F_2 : S \times \{0,1\}^L \to \pi_2$.

Исход. сост.	Сост. перех.	Входной сигнал
1	d_1 d_2 d_3	x_1 $\bar{x}_1 x_3$ $\bar{x}_1 \bar{x}_3$
2	d_2 d_3	x_2 $\bar{x}_2$
3	d_1 d_2 d_3	$\bar{x}_4$ $x_3 x_4$ $\bar{x}_3 x_4$
4	d_1 d_2 d_3	$\bar{x}_4$ $x_3 x_4$ $\bar{x}_3 x_4$
5	d_1 d_3	$\bar{x}_4$ x_4
6	d_1 d_3	$\bar{x}_4$ x_4
7	d_1 d_2 d_3	x_1 $\bar{x}_1 x_3$ $\bar{x}_1 x_3$
8	d_2 d_3	x_2 $\bar{x}_2$

Рис. 4. Таблица функции $F_3 : S \times \{0,1\}^L \to \pi_3$.

В заключение отметим, что задача декомпозиции заданного МПА в сеть с разделением входных переменных [3, 6] сводится к построению множества σ-разбиений с тем ограничением, что при генерации первичных σ-разбиений множеств $\mathcal{K}$ для всех состояний одинаково.

Литература

I. Jakobson G., Keevallik A., Leis P. Some aspects of the construction of microprogram automata networks. - In: Proceedings of the Second International Symposium on Discrete Systems. Dresden, 1977, vol. 1, p. 120-128.

2. Кеэваллик А.Э., Судницын А.В. Использование методов декомпозиции конечных автоматов для построения сетей микропроцессоров. - В кн.: Прикладные аспекты теории автоматов. - Тр. Международного семинара. Болгария, Варна, 1979, т. I, с. 54-65.

3. Лейс П.Л., Судницын А.В. Один метод декомпозиционного синтеза микропрограммных автоматов на программируемых логических матрицах. - В кн.: Прикладные аспекты теории автоматов. - Тр. Международного семинара. Болгария, Варна, 1979, т. I, с. 253-262.

4. Hartmanis J., Stearns R.E. Algebraic structure theory of sequential machines. - N.-Y., Prentice-Hall Inc., 1966. 209 p.

5. Баранов С.И. Синтез микропрограммных автоматов (граф-схемы и автоматы). Л.: Энергия, 1979. 232 с.

6. Беркман Б.Е. Метод выбора разбиений для декомпозиции МПА с разделением входных переменных. - Тр. Таллинск. политехн. ин-та, 1982, № 530, с. 93-105.

A. Sudnitsyn, V. Viies

The Synthesis of Discrete Control Automata Using Decomposition: the Partition Determination Problem

Abstract

A method for the decomposition of the microprogram automata into network of component automata with restricted number of essential binary input variables is introduced. The necessary and sufficient conditions for the automata decomposition existence are defined.

№ 626

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 62-507

Б.Е. Беркман

КОДИРОВАНИЕ ПРИ ПАРАЛЛЕЛЬНОЙ ПЕРЕДАЧЕ ДАННЫХ И ЕГО ИНФОРМАЦИОННЫЙ КРИТЕРИЙ

I. Введение. В различных задачах логического проектирования, в частности, связанных с декомпозиционными методами [I, 2], возникает задача двоичного кодирования разбиений, заданных на множестве равновероятных событий.

Одна из разновидностей такой задачи - кодирование аргументов и значений дискретных функций, которое направлено на упрощение булевых функций, выражающих зависимость отдельных битов значения дискретной функции от битов ее аргументов.

При оценке того, какой информацией обладает одно разбиение относительно другого, обычно используется понятие энтропии. В частности, понятие условной энтропии может использоваться при решении задач кодирования разбиений для оценки того, какое минимальное количество информации необходимо иметь, чтобы на основе этой информации и имеющегося разбиения получить полную информацию о другом разбиении [7, 8]. Энтропия является точной нижней оценкой средней длины кодовой комбинации при кодировании разбиений для случая последовательной передачи информации (когда отдельные биты передаются по одному каналу последовательно во времени) [3, 4]. Однако при параллельной передаче данных, когда отдельные биты передаются одновременно по нескольким двоичным каналам, при оценке количества требуемых каналов существенна не средняя, а максимальная длина комбинации. Поэтому при параллельной передаче данных необходимы другие методы и критерии кодирования, отличные от энтропийного. Настоящая статья посвящена проблемам поиска таких методов и критериев.

2. Кодирование в базисе (0,I). Разбиение произвольного множества $S=\{s_1,..,s_m\}$ на блоки $B_i^{(1)},\dots,B_i^{(\alpha)},\dots,B_i^{(m_i)}$, обозначим через $\pi_i(S)$. Количество блоков m_i в разбиении $\pi_i(S)$ обозначим через $|\pi_i|$. Нулевое и единичное разбиения обозначим через 0_S и 1_S, соответственно.

Каждому блоку $B^{(\alpha)}$ разбиения π поставим в соответствие двоичную кодовую комбинацию $L_\alpha=(\ell_\alpha^1,\dots,\ell_\alpha^r,\dots,\ell_\alpha^q)$, где $\ell_\alpha^r\in\{0,1\}$ $(r\in Q)$. Через Q будем обозначать $\{1,\dots,q\}$; q – длина кодовых комбинаций (одинаковая для всех блоков разбиения π).

Будем называть множество $\Lambda=\{L_\alpha|\alpha=1,..,|\pi|\}$ кодировкой разбиения π , если и только если $(L_\alpha=L_\beta)\Rightarrow(\alpha=\beta)$ (будем говорить, что все кодовые комбинации в кодировке уникальны).

Процедуру построения кодировки Λ для разбиения π будем называть кодированием разбиения π. Ясно, что условие $q\geqslant]\log_2|\pi|[$ – необходимое и достаточное условие существования для разбиения π кодировки Λ с длиной кодовой комбинации q (под $]x[$ будем понимать наименьшее целое, большее или равное x).

Определим систему двублочных-кодовых разбиений, соответствующих кодировке Λ разбиения π, следующим образом:

$$\pi_r=\{B_r^{(0)},B_r^{(1)}\,|\,B_r^0=\bigcup_{\ell_\alpha^r=0}B^{(\alpha)};\ B_r^{(1)}=\bigcup_{\ell_\alpha^r=1}B^{(\alpha)}\}.\ (r\in Q).$$

Из определения кодовых разбиений следует, что для любой кодировки Λ имеет место $\prod_{r=1}^{q}\pi_r=\pi$.

Можно показать также, что если задано множество двублочных разбиений $\{\pi_1,\dots,\pi_q\}$, произведение которых равно π , то у разбиения π существует кодировка Λ , кодовыми разбиениями для которой являются разбиения $\pi_1,\dots,\pi_q$. Построение такой кодировки не составляет труда.

Решением задачи построения для разбиения π кодировки Λ с заданными свойствами кодовых разбиений может явиться алгоритм построения цепочки разбиений $1_S=\eta_0,\eta_1,\dots,\eta_r,\dots,\eta_q=\pi$, определенной через $\eta_r=\eta_{r-1}\cdot\pi_r\,(r\in Q)$, а двублочные разбиения π_r обладают заданными свойствами.

3. Покрытия и полные покрытия. Следуя [I], определим алгебру покрытий.

Под покрытием $\tau_i(S)$ на множестве S будем понимать множество $\{B_i^{(1)},\dots,B_i^{(\alpha)},\dots,B_i^{(m_i)}\}$, для которого $\bigcup_{\alpha=1}^{m_i} B_i^{(\alpha)} = S$ и $(B_i^{(\alpha)} \subseteq B_i^{(\beta)}) \Rightarrow (\alpha = \beta)$.

Покрытие $\tau_1(S)$ больше или равно покрытию $\tau_2(S)$ (записывается $\tau_1 \geqslant \tau_2$), если и только если $(\forall B_2^{(\beta)})(\exists B_1^{(\alpha)})(B_2^{(\beta)} \subseteq B_1^{(\alpha)})$.

Для произвольного множества $\{S_t \mid S_t \subseteq S;\ t=1,\dots,n\}$ определим $\mathrm{Max}\{S_t\} = \{B \subseteq S \mid ((\exists t)(S_t = B)) \,\&\, ((B \subseteq S_t) \Rightarrow (B = S_t))\}$.

Произведение покрытий $\tau_1(S)$ и $\tau_2(S)$ – покрытие $\tau_3(S)$ такое, что

$$\tau_3 = \tau_1 \cdot \tau_2 = \mathrm{Max}\{B_1^{(\alpha)} \cap B_2^{(\beta)}\}.$$

Сумма покрытий $\tau_1(S)$ и $\tau_2(S)$ – покрытие $\tau_3(S)$ такое, что

$$\tau_3 = \tau_1 + \tau_2 = \mathrm{Max}(\tau_1 \cup \tau_2).$$

Отметим, что множество покрытий на S образует дистрибутивную решетку с операциями $\cdot$ и $+$ и порядком $\geqslant$ [1].

Как известно [5], каждому покрытию $\tau(S)$ соответствует отношение толерантности (рефлексивное и симметричное) $T_\tau(S)$. В то же время, каждому отношению толерантности $T(S)$ соответствует одно или несколько покрытий. Будем называть покрытия, которым соответствует одно и то же отношение $T(S)$, эквивалентными (обозначается $\tau_1 \sim \tau_2$).

Определение 1. Подмножество S' множества S называется кликой отношения толерантности $T(S)$, если $(\forall s_t \in S \setminus S')(\exists s_p \in S')(\neg T(s_t, s_p)) \,\&\, (\forall s_t \in S')(\forall s_p \in S')(T(s_t, s_p))$.

Определение 2. Покрытие $\tau(S)$ называется полным, если оно является множеством всех клик некоторого отношения толерантности $T(S)$.

Очевидно, что имеется взаимно однозначное соответствие между множеством всех полных покрытий на S и множеством отношений толерантности на S, и среди эквивалентных покрытий, соответствующих любому отношению $T(S)$, имеется одно полное покрытие, равное сумме всех эквивалентных покрытий. Заметим, что одноблочное (единичное) и любое двублочное покрытие – полные, и если покрытие является разбиением, то оно полное.

Заметим также, что задача нахождения полного покрытия, эквивалентного заданному покрытию (или соответствующего заданному отношению $T(S)$), будет являться NP – полной [9]. Бу-

дем обозначать полное покрытие, эквивалентное покрытию $\tau_i(S)$, через $\tilde{\tau}_i(S)$.

Теорема I. Если $\tau_1(S)$ и $\tau_2(S)$ – полные покрытия и $\tau_3(S)=\tau_1(S)\cdot\tau_2(S)$, то $\tau_3(S)$ – полное покрытие.

Доказательство. Проведем доказательство от противного. Допустим, что покрытие τ_3 – неполное. Тогда

$$(\exists B_3^{(\alpha)})(\exists s_t\in S\setminus B_3^{(\alpha)})(\forall s_p\in B_3^{(\alpha)})(s_t T_{\tau_3} s_p).$$

Поскольку $T_{\tau_3}=T_{\tau_1}\cap T_{\tau_2}$, то $s_t T_{\tau_3} s_p \Rightarrow s_t T_{\tau_1} s_p \,\&\, s_t T_{\tau_2} s_p$.

Отсюда, в отношениях T_{τ_1} и T_{τ_2} имеются клики, включающие $(B_3^{(\alpha)}\cup\{s_t\})$. Из этого следует (поскольку τ_1 и τ_2 – полные покрытия), что $(\exists B_1^{(\beta)})(\exists B_2^{(\gamma)})((B_3^{(\alpha)}\cup\{s_t\}\subseteq B_1^{(\beta)})\,\&\,B_3^{(\alpha)}\cup\{s_t\}\subseteq B_2^{(\gamma)}))$.

Тогда $(B_3^{(\alpha)}\cup\{s_t\})\subseteq(B_1^{(\beta)}\cap B_2^{(\gamma)})$. С другой стороны, поскольку $\tau_3=\tau_1\cdot\tau_2$, $(\exists B_3^{(\alpha')}\in\tau_3)(B_1^{(\beta)}\cap B_2^{(\gamma)}\subseteq B_3^{(\alpha')})$.

Следовательно, $(B_3^{(\alpha)}\cup\{s_t\})\subseteq B_3^{(\alpha')}$. По определению покрытия,

$$(B_3^{(\alpha)}\subseteq B_3^{(\alpha')})\Rightarrow(\alpha=\alpha').$$

Отсюда $B_3^{(\alpha)}\cup\{s_t\}\subseteq B_3^{(\alpha)}$, то есть $s_t\in B_3^{(\alpha)}$, что противоречит исходной посылке $s_t\in S\setminus B_3^{(\alpha)}$. Таким образом, τ_3 – полное покрытие.

Можно показать, что в отличие от произведения, сумма полных покрытий не обязательно является полным покрытием. Однако можно определить сумму $\oplus$ полных покрытий $\tau_1\oplus\tau_2 \overset{Def}{=} (\widetilde{\tau_1+\tau_2})$.

Легко показать, что для любых полных покрытий τ_1,τ_2 и τ_3 имеет место $\tau_3=\tau_1\cdot\tau_2\Leftrightarrow T_{\tau_3}=T_{\tau_1}\cap T_{\tau_2}$ и $\tau_3=\tau_1\oplus\tau_2\Leftrightarrow T_{\tau_3}=T_{\tau_1}\cup T_{\tau_2}$ (здесь $\cap$ и $\cup$ – операции пересечения и объединения отношений [5]). Отсюда следует, что множество полных покрытий на S образует дистрибутивную решетку с операциями $\cdot$ и $\oplus$ и порядком $\geqslant$, изоморфную дистрибутивной решетке отношений толерантности на S.

4. Кодирование в базисе (0,I,–). Задано разбиение π на множестве S. Каждому блоку $B^{(\alpha)}(\alpha=1,\ldots,|\pi|)$ ставится в соответствие кодовая комбинация $M_\alpha=(M_\alpha^1,\ldots,M_\alpha^r,\ldots,M_\alpha^q)$, компоненты которой $M_\alpha^r\in\{0,I,-\}(r\in Q)$. В отличие от двоичных кодовых комбинаций L_α, определенных в базисе (0,I), будем называть M_α троичными кодовыми комбинациями или интервалами [6].

Каждому интервалу M_α будет соответствовать множество порожденных двоичных комбинаций $G_\alpha=\{L_{\alpha x}\mid x=1,\dots,y\}$, где y – количество всех порожденных комбинаций для M_α; $L_{\alpha x}=(l^1_{\alpha x},\dots,l^r_{\alpha x},\dots,l^q_{\alpha x})$, причем для любых $x=1,\dots,y$ и $r\in Q$ верно

$$\begin{cases} M^r_\alpha=0 \Rightarrow l^r_{\alpha x}=0, \\ M^r_\alpha=1 \Rightarrow l^r_{\alpha x}=1, \\ M^r_\alpha=- \Rightarrow (l^r_{\alpha x}=0 \vee l^r_{\alpha x}=1). \end{cases}$$

Двоичные комбинации $L_{\alpha x}$ соответствуют точкам булева пространства, входящим в интервал M_α.

Множество $\Omega=\{M_\alpha\mid\alpha=1,\dots,|\pi|\}$ будем называть (непротиворечивой) троичной кодировкой разбиения π, если интервалы булева пространства M_α и $M_\beta(M_\alpha,M_\beta\in\Omega)$ попарно не пересекаются (ортогональны) [6], то есть $(\alpha\neq\beta)\Rightarrow(G_\alpha\cap G_\beta=\phi)$.

Непротиворечивость троичной кодировки означает, что любая двоичная комбинация $(l^1,\dots,l^r,\dots,l^q)$ будет однозначно определять блок $B^{(\alpha)}$ в разбиении π, в интервал M_α для которого эта комбинация входит.

Будем называть интервалы M_α и M_β неразличимыми по r-й компоненте, если $(M^r_\alpha=M^r_\beta)\vee(M^r_\alpha=-)\vee(M^r_\beta=-)$.

Будем называть интервалы неразличимыми по совокупности компонент с номерами $(r_1,\dots,r_\alpha)$, если они неразличимы по каждой из этих компонент. Свойство, обратное неразличимости, назовем различимостью.

Очевидно, что условие непротиворечивости кодировки совпадает с условием различимости любой пары интервалов M_α и M_β по совокупности всех компонент из Q.

Определим множество двублочных кодовых покрытий $\{\tau_r\mid r\in Q\}$, соответствующее кодировке Ω

$$\tau_r=\{B^{(0)}_r,B^{(1)}_r\mid B^{(0)}_r=\bigcup_{M^r_\alpha\in\{0,-\}}B^{(\alpha)};\quad B^1_r=\bigcup_{M^r_\alpha\in\{1,-\}}B^{(\alpha)}\},$$

то есть в один блок τ_r объединяются все блоки π, интервалы для которых неразличимы по r-й компоненте.

Теорема 2. Множество двублочных покрытий $\{\tau_1,\dots,\tau_q\}$ является множеством кодовых покрытий для некоторой кодировки Ω разбиения π, если и только если $\prod_{r=1}^{q}\tau_r=\pi$ (I).

Доказательство. Достаточность. Пусть имеется множество двублочных покрытий $\{\tau_1,\ldots,\tau_q\}$ таких, что $\prod_{r=1}^{q}\tau_r=\pi$. Покажем, как построить кодировку Ω разбиения π с множеством кодовых покрытий $\{\tau_1,\ldots,\tau_q\}$.

Пусть $\tau_r=\{B_r^{(0)},B_r^{(1)}\}(r\in Q)$. Тогда по условию (I) для любых $r\in Q$ и $B^{(\alpha)}\in\pi$ справедливо $(B^{(\alpha)}\subseteq B_r^{(0)}\vee B^{(\alpha)}\subseteq B_r^{(1)})$.

Поставим в соответствие каждому блоку $B^{(\alpha)}$ разбиения π кодовую комбинацию M_α такую, что для любого $r\in Q$ имеем

$$M_\alpha^r=\begin{cases}0, & \text{если } B^{(\alpha)}\subseteq B_r^{(0)}\ \&\ \neg(B^{(\alpha)}\subseteq B_r^{(1)}),\\ I, & \text{если } B^{(\alpha)}\subseteq B_r^{(1)}\ \&\ \neg(B^{(\alpha)}\subseteq B_r^{(0)}),\\ -, & \text{если } B^{(\alpha)}\subseteq B_r^{(0)}\cap B_r^{(1)}.\end{cases}$$

Из условия (I) следует, что для каждых различных $B^{(\alpha)}\in\pi$ и $B^{(\beta)}\in\pi$ существует $r\in Q$ такой, что $((B^{(\alpha)}\cup B^{(\beta)})\not\subseteq B_r^{(0)})\&((B^{(\alpha)}\cup B^{(\beta)})\not\subseteq B_r^{(1)})$.

Это означает, что любая пара определенных нами комбинаций M_α и M_β будет различима по некоторой $(r-й)$ компоненте и полученное множество $\Omega=\{M_\alpha\mid\alpha\in\{1,\ldots,|\pi|\}\}$ будет кодировкой.

Необходимость. Доказательство может быть проведено от противного и для краткости опускается.

На основе теоремы 2 можно рассматривать задачу построения для разбиения π кодировки Ω с заданными свойствами кодовых покрытий как задачу построения цепочки $1_S=\eta_0,\eta_1,\ldots,\eta_r,\ldots,\eta_q=\pi$, определенной через $\eta_r=\eta_{r-1}\cdot\tau_r(r\in Q)$, где двублочные покрытия τ_r обладают заданными свойствами. Поскольку покрытия τ_r полные (т.к. они двублочные), из теоремы I следует, что все элементы цепочки η_r также полные покрытия.

Заметим, что кодирование двоичными комбинациями различной длины, используемое для последовательной передачи данных, соответствует частному случаю кодирования в базисе $(0,I,-)$. При этом длина q_γ троичных кодовых комбинаций равна максимальной длине $\max(q_\alpha)$ двоичных, а каждой α-й двоичной комбинации длины q_α соответствует троичная, в которой первые q_α битов совпадают с соответствующими битами двоичной комбинации, а оставшиеся $(q_\gamma-q_\alpha)$ битов равны $-$.

5. Ресурс информации. При оценке длины двоичной кодовой комбинации, требуемой для кодирования разбиения $\pi(S)=\{B^{(\alpha)}\mid\alpha=1,\ldots,k\}$, может использоваться понятие энтропии

$$H(\pi) = -\sum_{\alpha=1}^{k} \frac{n_\alpha}{m} \cdot \log_2 \frac{n_\alpha}{m},$$

где $m = |S|$; $n_\alpha = |B^{(\alpha)}|$.

В соответствии с известным результатом теории информации [3, 4] имеется двоичная кодировка разбиения π (возможно, с разной длиной кодовых комбинаций), средняя длина $\tilde{L}$ которой удовлетворяет неравенству $H(\pi) \le \tilde{L} < H(\pi)+1$.

Под средней длиной $\tilde{L}$ кодовой комбинации понимается $\sum_{\alpha=1}^{k} \frac{n_\alpha}{m} \cdot \log_2 q_\alpha$ (q_α - длина комбинации, соответствующей блоку $B^{(\alpha)}$ разбиения π).

Энтропийный критерий $H(\pi)$ и понятие средней длины $\tilde{L}$ отражают эффективность кодирования [4] и среднее количество бит, требуемое для кодирования, для случая последовательной передачи информации (когда отдельные биты кодовых комбинаций передаются по одному каналу последовательно во времени).

При параллельной передаче информации, используемой в локальных системах (когда каждый бит кодовой комбинации передается по отдельному двоичному каналу), требуемый ресурс (количество двоичных каналов) определяется максимальной длиной кодовой комбинации.

Минимальное количество двоичных каналов, требуемое для параллельной передачи кода разбиения π, равно $L =]\log_2 k[$ ($k = |\pi|$).

Энтропия разбиения $H(\pi)$ меньше или равна $\log_2 k$ и достигает значения $\log_2 k$ только для случая $n_\alpha = \frac{m}{k}$ ($\alpha = 1,\ldots,k$) [4], то есть при равных размерах блоков разбиения π.

В случае неравномерных разбиений π (когда размеры блоков сильно различаются) оценка $H(\pi)$ будет давать результаты, заметно занижающие действительно требуемый при параллельной передаче ресурс.

Пример. Если $\pi = \{B^{(1)}, B^{(2)}, \ldots, B^{(16)}\}$, где $B^{(1)} = \overline{1,2,\ldots,1009}$; $B^{(2)} = 1010$; $B^{(3)} = 1011$; $B^{(4)} = 1012$; ...; $B^{(16)} = 1024$, то энтропия $H(\pi) = \frac{-1}{1024}\,(15 \cdot 1 \cdot \log_2 \frac{1}{1024} + 1009 \cdot \log_2 \frac{1009}{1024}) \approx 0.16$.

В то же время, минимальное количество двоичных каналов (битов) для параллельной передачи кода разбиения π равно $L = \log_2 16 = 4$.

Таким же недостатком (заниженность оценки) обладает и условная энтропия разбиений $H(\pi_2/\pi_1) = H(\pi_1 \cdot \pi_2) - H(\pi_1)$, когда она используется для оценки минимального количества двоичных каналов при параллельной передаче информации, которой обладает разбиение π_2 относительно разбиения π_1.

В настоящей статье предлагается информационный критерий, позволяющий точно оценить минимальное количество двоичных каналов (битов), требуемое для того, чтобы переданная по ним двоичная комбинация при известном номере блока разбиения π_1 однозначно определяла номер блока разбиения π_2. Более того, введенный критерий позволяет также оценить (с точки зрения параллельной передачи) информацию, содержащуюся в одном покрытии относительно другого.

Определение 3. Под инфоресурсом $R(\tau_2/\tau_1)$ покрытия $\tau_2(S)$ по отношению к $\tau_1(S)$ будем понимать следующую величину $R(\tau_2/\tau_1) = \log_2(\min(|\pi_3|))$, где $\pi_3(S)$ - разбиение, для которого $\tilde{\tau}_1 \cdot \pi_3 \le \tilde{\tau}_2$; минимум берется по всем таким разбиениям.

Очевидно, что $]R(\tau_2/\tau_1)[$ точно равно минимальному количеству двублочных разбиений (покрытий), произведение которых друг на друга и на покрытие $\tilde{\tau}_1$ будет меньше или равно $\tilde{\tau}_2$.

Следуя [7], где определено сужение разбиений, назовем сужением покрытия $\tau_i(S)$ на $S' \subset S$ систему

$$\bar{\tau}_i(S') = \{B_i^{(\alpha)} \cap S' \mid (B_i^{(\alpha)} \in \tau_i) \& (B_i^{(\alpha)} \cap S' \neq \phi)\}.$$

Тогда $|\bar{\tau}_i(S')|$ равно количеству блоков покрытия τ_i, пересечение которых с S' не пусто.

Теорема 3. Для двух разбиений $\pi_1(S)$ и $\pi_2(S)$

$$R(\pi_2/\pi_1) = \log_2(\max_{B_1^{(\alpha)} \in \pi_1} |\bar{\pi}_2(B_1^{(\alpha)})|).$$

Доказательство. Обозначим $\max_{B_1^{(\alpha)} \in \pi_1} |\bar{\pi}_2(B_1^{(\alpha)})|$ через k^*. Требуется показать, что $R(\pi_2/\pi_1) = \log_2 k^*$.

Покажем сначала, что существует разбиение $\pi_3(S)$ такое, что

$$(\pi_1 \cdot \pi_3 \le \pi_2) \& (|\pi_3| = k^*).$$

Построим множество $\{\bar{\pi}_2(B_1^{(\alpha)}) \mid B_1^{(\alpha)} \in \pi_1\}$ сужений разбиения π_2 на блоках разбиения π_1. Блоки разбиений $\bar{\pi}_2(B_1^{(\alpha)})$ $(\alpha = 1, \ldots, |\pi_1|)$

обозначим через $\bar{B}_2^{(\beta)}(B_1^{(\alpha)})(\beta=1,\ldots,|\bar{\pi}_2(B_1^{(\alpha)})|)$. Примем, что $\bar{B}_2^{(\beta)}(B_1^{(\alpha)})=\emptyset$, если $|\bar{\pi}_2(B_1^{(\alpha)})|<\beta\leq k^*$.

Построим теперь разбиение $\pi_3(S)$ следующим образом:

$$\pi_3=\{B_3^{(\beta)}\mid B_3^{(\beta)}=\bigcup_{B_1^{(\alpha)}\in\pi_1}\bar{B}_2^{(\beta)}(B_1^{(\alpha)});\ \beta=1,\ldots,k^*\},$$

По способу построения разбиения π_3 ясно, что

$$\pi_1\cdot\pi_3=\bigcup_{B_1^{(\alpha)}\in\pi_1}\bar{\pi}_2(B_1^{(\alpha)})=\pi_1\cdot\pi_2\leq\pi_2.$$

Теперь покажем, что для любого разбиения $\pi_3(S)$ такого, что $\pi_1\cdot\pi_3\leq\pi_2$, имеет место $|\pi_3|\geq k^*$. Рассмотрим блок $B_1^{(\alpha)}$, для которого $|\bar{\pi}_2(B_1^{(\alpha)})|=k^*$. Выберем по одному состоянию s^β из каждого блока $\bar{B}_2^{(\beta)}$ разбиения $\bar{\pi}_2(B_1^{(\alpha)})(\beta=1,\ldots,k^*)$. Любая пара s^β и s^γ из выбранных состояний принадлежит блоку $B_1^{(\alpha)}$ в разбиении π_1, но разным блокам в разбиении π_2. Из этого (с учетом условия $\pi_1\cdot\pi_3\leq\pi_2$) вытекает, что любая пара s^β и s^γ из выбранных состояний должна принадлежать разным блокам в разбиении π_3. Отсюда следует, что количество блоков в разбиении π_3 должно быть не меньше k^*. Теорема доказана.

Можно показать (аналогично второй части доказательства теоремы 3), что для любых покрытия $\tau_1(S)$ и разбиения $\pi_2(S)$ верно

$$R(\pi_2/\tau_1)\geq\log_2(\max_{B_1^{(\alpha)}\in\tilde{\tau}_1}|\bar{\pi}_2(B_1^{(\alpha)})|);\quad R(\tau_1/\pi_2)\leq\log_2(\max_{B_2^{(\alpha)}\in\pi_2}|\tilde{\tilde{\tau}}_1(B_2^{(\alpha)})|).$$

Для определения точного значения $R(\tau_2/\tau_1)$ в общем случае рассмотрим неравенство $\tilde{\tau}_1\cdot\pi_3\leq\tilde{\tau}_2$. Обозначим отношения толерантности, соответствующие τ_1 и τ_2, через T_1 и T_2, соответственно, а отношение эквивалентности, соответствующее π_3, через E_3. Тогда

$$(\tilde{\tau}_1\cdot\pi_3\leq\tilde{\tau}_2)\Leftrightarrow(T_1\cap E_3\subseteq T_2)\Leftrightarrow$$
$$\Leftrightarrow(T_1\cap E_3\cap\neg T_2\subseteq T_2\cap\neg T_2)\Leftrightarrow(T_1\cap\neg T_2)\cap E_3=\phi,$$

где ϕ - нуль-отношение, а $\neg T_2$ - дополнение отношения $T_2(S)$ до универсального отношения $S\times S$.

Можно показать, что условие $(T_1\cap\neg T_2)\cap E_3=\phi$ эквивалентно условию "π_3 является множеством одноцветных классов для раскраски [5] графа отношения $(T_1\cap\neg T_2)$".

Отсюда точное значение $R(\tau_2/\tau_1)$ для произвольных покрытий τ_1 и τ_2 равно хроматическому числу [5] графа отношения $T_1\cap\neg T_2$.

Несложно показать, что инфоресурс обладает следующими свойствами:

1. $(\tilde{\tau}_1 \leq \tilde{\tau}_2) \Leftrightarrow R(\tau_2/\tau_1) = 0$.

2. $((\tilde{\tau}_1 \leq \tilde{\tau}_2) \,\&\, (\tilde{\tau}_3 \geq \tilde{\tau}_4)) \Rightarrow R(\tau_3/\tau_1) \leq R(\tau_4/\tau_2)$.

3. Если τ_3 - двублочное покрытие, то $R(\tau_2/\tau_1 \cdot \tau_3) \geq$ $\geq R(\tau_2/\tau_1) - 1$.

4. $R(\tau_2/\tau_1) \leq R(\tau_2/1_S) \leq \log_2 |\tilde{\tau}_2|$.

5. $R(\tau_2/\tau_1) = R(\tau_1 \cdot \tau_2/\tau_1) = R(\tau_2/\tau_1 + \tau_2)$.

6. $R(\tau_3/\tau_1) \leq R(\tau_3/\tau_2) + R(\tau_2/\tau_1)$.

7. Условие $((\tilde{\tau}_2 \leq \tilde{\tau}_1) \& (\exists R(\tau_2/\tau_1)[\leq q]))$ - необходимое и достаточное условие существования цепочки $\tilde{\tau}_1 = \eta_0, \eta_1 \ldots, \eta_r, \ldots, \eta_q = \tilde{\tau}_2$, где $\eta_r = \eta_{r-1} \cdot \theta_r$; θ_r - некоторые двублочные покрытия $(r \in Q)$.

6. Использование инфоресурса. Определенное нами понятие инфоресурса может быть использовано в различных задачах, связанных с выбором и кодированием разбиений, в частности - в задачах декомпозиции автоматов методом пар разбиений [1, 2]. Точные описания алгоритмов решения этих задач выходят за рамки настоящей статьи.

Покажем, как может быть применен инфоресурс для кодирования в базисе (0,1,-) аргументов и значений дискретных функций.

Идею этого применения будем демонстрировать на примере кодирования аргументов и значений дискретной функции $F: \pi' \times \pi'' \times \pi''' \to \pi$, заданной табл. 1. В нашем примере

$\pi' = \overline{1,2,3,4};\ \overline{5,6,7};\ \overline{8,9};\ \overline{10,11,12} = \{a_1, a_2, a_3, a_4\}$

$\pi'' = \overline{\overline{1,2,3,5,6,8,9};\ \overline{4,7,10,11};\ \overline{12}} = \{b_1, b_2, b_3\}$

$\pi''' = \overline{\overline{1,2,4,10};\ \overline{5,8};\ \overline{3,6};\ \overline{7,9,11,12}} = \{c_1, c_2, c_3, c_4\}$

$\pi = \overline{\overline{1,2};\ \overline{3};\ \overline{4};\ \overline{5,6};\ \overline{7};\ \overline{8};\ \overline{9};\ \overline{10};\ \overline{11};\ \overline{12}};$ $= \{d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\}$.

Задаемся длинами кодовых комбинаций и обозначениями булевых переменных, соответствующих отдельным компонентам кодовых комбинаций для наших разбиений.

$\pi' - (q' = 2;$ биты $x_1, x_2);$ $\pi'' - (q'' = 2;$ биты $x_3, x_4);$

$\pi''' - (q''' = 2;$ биты $x_5, x_6);$ $\pi - (q = 4;$ биты $y_1, y_2, y_3, y_4)$.

Если произвести простое ненаправленное кодирование раз-

биений (a_1 - 00, a_2 - 01, a_3 - 10, a_4 - 11; b_1 - 00, b_2 - 01, b_3 - 10; c_1 - 00, c_2 - 01, c_3 - 10, c_4 - 11; d_1 - 0000, d_2 - - 0001, ..., d_9 - 1000, d_{10} - 1001), то после выделения булевых функций $y_1=f_1(x_1,\ldots,x_6);\ldots; y_4=f_4(x_1,\ldots,x_6)$ и их минимизации методом конкурирующих интервалов [6], получаем следующее приближенно минимальное решение:

$$\begin{cases} y_1 = x_1 x_2 x_6; \\ y_2 = x_1 \bar{x}_5 \vee x_1 \bar{x}_2 \vee x_2 x_4 x_6; \\ y_3 = x_4 \bar{x}_6 \vee x_2 \bar{x}_4 \vee x_1 \bar{x}_2 x_5; \\ y_4 = x_2 x_3 \vee x_2 \bar{x}_6 \vee x_5 \bar{x}_6 \vee \bar{x}_5 x_6. \end{cases}$$

Теперь проведем направленное кодирование разбиений, используя для выбора кодовых разбиений (покрытий) понятие инфоресурса.

Кодирование будем производить, исходя из следующих соображений:

I°. Кодовые цепочки $1_S = \eta_0, \eta_1, \eta_2, \eta_3, \eta_4 = \pi$;

$1_S = \eta'_0, \eta'_1, \eta'_2 = \pi'$;

$1_S = \eta''_0, \eta''_1, \eta''_2 = \pi''$;

$1_S = \eta'''_0, \eta'''_1, \eta'''_2 = \pi'''$

будем строить параллельно.

Таблица I

Дискретная функция $F: \pi' \times \pi'' \times \pi''' \to \pi$

π'	π''	π'''	π
a_1	b_1	c_1	d_1
a_1	b_1	c_3	d_2
a_1	b_2	c_1	d_3
a_2	b_1	c_2	d_4
a_2	b_1	c_3	d_4
a_2	b_2	c_4	d_5
a_3	b_1	c_2	d_6
a_3	b_1	c_4	d_7
a_4	b_2	c_1	d_8
a_4	b_2	c_4	d_8
a_4	b_3	c_4	d_{10}

2^0. Покрытие τ_r для любой кодовой цепочки $1_S = \eta_0, \ldots, \eta_r, \ldots, \eta_q = \pi$ должно (по свойству 7 инфоресурса) удовлетворять условиям

$$\left.\begin{array}{l} \tau_r \geq \pi; \\ R(\pi/\eta_r) = R(\pi/\eta_{r-1}\cdot\tau_r) \leq q-r \end{array}\right\} \qquad (r \in Q).$$

3^0. Для уменьшения зависимости битов (y_1, y_2, y_3, y_4) значения функции F от битов ее аргументов $(x_1, x_2, \ldots, x_6)$ необходимо минимизировать количества аргументов k_r в функциях

$$f_r : \underset{j=1,\ldots,k_r}{\times} \theta_j \to \tau_r \quad (r = 1, \ldots, 4).$$

Здесь θ_j – двублочные покрытия из множества $\{\tau_1', \tau_2', \tau_1'', \tau_2'', \tau_1''', \tau_2'''\}$ кодовых покрытий для разбиений π', π'', π''', а τ_r – кодовое покрытие для разбиения π.

Поэтому при выборе кодовых покрытий будем стремиться минимизировать количества элементов k_r в произведениях

$\prod_{j=1}^{k_r} \theta_j$, где $\prod_{j=1}^{k_r} \theta_j \leq \tau_r$.

В нашем примере сначала выбираем

$\tau_1 = \tau_1' = \overline{1,2,3,4,8,9;\ 5,6,7,10,11,12}$ $\qquad (k_1 = 1);$

$R(\pi/\eta_1) = R(\pi/\eta_0\cdot\tau_1) = \log_2 5 \approx 2.32 \leq q-1 = 3;$

$R(\pi'/\eta_1') = R(\pi'/\eta_0'\cdot\tau_1') = \log_2 2 = 1 \leq q'-1 = 1.$

Затем аналогично выбираем

$\tau_2 = \tau_2' = \overline{1,2,3,4,5,6,7;\ 8,9,10,11,12}$ $\qquad (k_2 = 1);$

$\tau_3 = \tau_1' = \overline{1,2,3,5,6,8,9,12;\ 4,7,10,11}$ $\qquad (k_3 = 1);$

$\tau_4 = \overline{1,2,4,5,6,8,10;\ 3,5,6,7,9,11,12} \geq \tau_1'' = \overline{1,2,4,10,5,8;\ 3,6,7,9,11,12}$ $\quad (k_4 = 1).$

Завершая кодирование π'' и π''' выбираем

$\tau_2'' = \overline{1,2,3,4,5,6,7,8,9,10,11;\ 12};$

$\tau_2''' = \overline{1,2,3,4,6,10;\ 5,7,8,9,11,12}.$

Получили следующие кодировки

– для π' (биты x_1, x_2 на основе τ_1' и τ_2'):

$a_1 - 00;\ a_2 - 10;\ a_3 - 01;\ a_4 - 11.$

– для π'' (биты x_3, x_4 на основе τ_1'' и τ_2''):

b_1 - 00; b_2 - 10; b_3 - 01.

- для π''' (биты x_5, x_6 на основе τ_1''' и τ_2'''):

c_1 - 00; c_2 - 01; c_3 - 10; c_4 - 11.

- для π (биты y_1, y_2, y_3, y_4 на основе $\tau_1, \tau_2, \tau_3, \tau_4$):

d_1 - 0000; d_2 - 0001; d_3 - 0010; d_4 - 1000; d_5 - 1011;
d_6 - 0100; d_7 - 0101; d_8 - 1110; d_9 - 1111; d_{10} - 1101.

Уже из процедуры кодирования видно, что в результате каждый бит значения функции зависит от одного бита аргументов ($y_1 = x_1$; $y_2 = x_2$; $y_3 = x_3$; $y_4 = x_5$).

Таким образом, кодирование на основе критерия инфоресурса позволило предельно упростить булевы функции, реализующие заданную дискретную функцию F.

7. Заключение. На основе аппарата покрытий и отношений исследованы вопросы кодирования в базисах (0,1) и (0,1,-) для случая параллельной передачи информации. Разработано понятие инфоресурса, являющегося точным информационным критерием кодирования при параллельной передаче данных, и показана целесообразность использования критерия инфоресурса для кодирования дискретных функций. Разработанное понятие инфоресурса применимо в различных задачах теории конечных автоматов, связанных с выбором и кодированием разбиений. Необходимы дальнейшие исследования для создания алгоритмов и программ выбора и кодирования разбиений на основе критерия инфоресурса.

Литература

1. Hartmanis J., Stearns R.E. Algebraic structure theory of sequential machines. - Englewood Cliffs, N.-Y., Prentice-Hall Inc., 1966, 211 p.

2. Jakobson G., Keevallik A., Leis P. Some aspects of the construction of microprogramm automata networks. - In: IFAC-Symposium "Discrete Systems". Dresden, 1977, p. 120-128.

3. Шеннон К. Математическая теория связи. - В сб.: Работы по теории информации и кибернетике. М.: ИЛ, 1963, с. 379-420.

4. Хемминг Р.В. Теория кодирования и теория информации. - М.: Радио и связь, 1983, с. 78-92.

5. О р е О. Теория графов. - М.: Наука, 1986. 336 с.

6. З а к р е в с к и й А.Д. Логический синтез каскадных схем. - М.: Наука, 1981. 416 с.

7. Л а у с м а а Т. Об алгебраических основах понятия энтропии. - Известия АН ЭССР, Физ. Матем., 1983, 32, № 2, с. 128-134.

8. Л а у с м а а Т. Об информационных свойствах разбиений. - Известия АН ЭССР, Физ. Матем., 1982, 31, № 4, с. 390.

9. Р е й н г о л ь д Э., Н и в е р н е л ь т Ю., Д е о Н. Комбинаторные алгоритмы. Теория и практика. - М.: Мир, 1980, с. 389-463.

B. Berkman

Informational Measure of Coding for Parallel Data Transmission

Abstract

In this paper coding for parallel data transmission is considered. The partition coding methods in bases (0,1) and (0,1,-) are developed. The concept of information resource is introduced. It is shown that the information resource is the exact measure of information for coding in terms of parallel data transmission whereas the concept of entropy gives a lower value in this case. An effective method is proposed, based on information resource for coding discrete functions, which are extensively used in the structure theory of finite automata.

№ 626

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 62-507

А. Кеэваллик, М. Круус

МЕТОД НАХОЖДЕНИЯ ДИАГНОСТИЧЕСКИХ И УСТАНОВОЧНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ДЛЯ СЕТЕЙ АВТОМАТОВ

Введение. В комплексе проблем, связанных с проведением контрольных экспериментов с конечными автоматами, одно из центральных мест занимают задачи определения начального и конечного состояния автомата [1-3]. Эти задачи решаются путем нахождения диагностических и установочных последовательностей (ДП и УП), позволяющие различать начальные и конечные состояния соответственно [4, 6].

В данной работе рассматриваются задачи нахождения ДП и УП для сетей автоматов, полученных в результате декомпозиции заданного автомата [5]. При этом сеть автоматов рассматривается как черный ящик и ДП и УП находятся отдельно для всех ее компонентных автоматов.

В работе выведены условия существования ДП и УП для компонентных автоматов, показывающих их взаимосвязь с процессом декомпозиции автомата. Предложен метод нахождения ДП и УП для сетей автоматов.

Основные понятия. Как обычно, конечный автомат Мили - это пятерка $A=(I,S,O,\delta,\lambda)$, где I - входной алфавит, S - множество внутренних состояний, O - выходной алфавит, $\delta: S\times I\to S$ - функция переходов и $\lambda: S\times I\to O$ - функция выходов. В дальнейшем считаем, что автомат сильносвязанный и минимальный.

Сеть автоматов [5] - это шестерка $N=(I,\{A_i\},O,\{f_i\},\{\psi_i\},g)$, где I - входной алфавит; $\{A_i=(I_i,S_i,\delta_i)\}$, $I_i=I_i'\times I_i''$, $1\le i\le n$, - множество компонентных автоматов; O - выходной алфавит; $\{f_i:(\times S_j)\to I_i'\}, 1\le i,j\le n$ - множество функций соединения; $\{\psi_i: I\to I_i''\}, 1\le i\le n$ - функции входных зависимостей; $g:(\times S_j)\times I\to O$ - функция выходов.

Сеть N – это математическая модель, описывающая функционирование совокупности взаимосвязанных (системы) автоматов. По сети N можно найти эквивалентный ей автомат $A_N = (I_N, S_N, O_N, \delta_N, \lambda_N)$, который называем результирующим автоматом сети N. Автомат A_N определяется следующим образом[5]:

$I_N = I$;

$S_N = \times S_i$, $1 \le i \le n$; $O_N = O$; $\delta_N : S_N \times I_N \to S_N$,

$\delta_N[(\sigma_1, \sigma_2, \ldots, \sigma_n), x] = (\delta_1[\sigma_1, f_1(\sigma_1, \sigma_2, \ldots, \sigma_n), \psi_1(x)]$,

$\delta_2[\sigma_2, f_2(\sigma_1, \sigma_2, \ldots, \sigma_n), \psi_2(x)], \ldots, \delta_n[\sigma_n, f_n(\sigma_1, \sigma_2, \ldots, \sigma_n), \psi_n(x)])$;

$\lambda_N : S_N \times I_N \to O_N$, $\lambda_N[(\sigma_1, \sigma_2, \ldots, \sigma_n), x] = g[(\sigma_1, \sigma_2, \ldots, \sigma_n), x]$.

Обозначим через $\pi p_i(\sigma_1, \sigma_2, \ldots, \sigma_n)$ i-й компонент вектора $(\sigma_1, \sigma_2, \ldots, \sigma_n)$.

Сеть N называем декомпозицией автомата A, если ее результирующий автомат A_N реализует A. При этом под реализацией понимаем следующее: автомат A_N реализует автомат A, если у него существует подавтомат, который изоморфен автомату A [5].

Поскольку предложенный в [5] и используемый в данной работе конструктивный метод декомпозиции опирается на аппарат разбиений, то приведем символику, которую используем для описания свойств разбиений и отношений между ними: π_i – разбиение множества S; $\pi_i(\sigma)$ – блок разбиения π_i, содержащий состояние σ; $\sigma \equiv t(\pi_i)$ – σ и t содержатся в одном и том же блоке разбиения π_i; $\sigma \equiv t(\pi_1 \cdot \pi_2) \Leftrightarrow \sigma \equiv t(\pi_1) \,\&\, \sigma \equiv t(\pi_2)$; $\pi_1 \le \pi_2 \Leftrightarrow \pi_1 \cdot \pi_2 = \pi_1$; 0 – поэлементное разбиение множества S; 1 – одноблочное разбиение множества S; система разбиений называется полной, если $\prod_{i=1}^{n} \pi_i = 0$.

В [5] доказано, что полнота системы разбиений $P = \{\pi_i\}$, $1 \le i \le n$, является необходимым и достаточным условием существования декомпозиции автомата A. При этом устанавливается взаимооднозначное соответствие между разбиениями π_i и компонентными автоматами A_i.

Приведем конструктивный способ построения сети N.

Поставим каждому разбиению π_i в соответствие функцию $F_i : S \times I \to \pi_i$, $F_i(\sigma, x) = \pi_i(\delta(\sigma, x))$. Образует на S и I разбиения τ_i и η_i следующим образом:

$$\sigma \equiv t(\tau_i) \Leftrightarrow \forall x \in I\,[F_i(\sigma, x) = F_i(t, x)];$$

$$x \equiv y(\eta_i) \Leftrightarrow \forall s \in S\,[F_i(s,x) = F_i(s,y)].$$

Построим автоматы $A_i = (I_i, S_i, \delta_i)$. Выбираем из множества $\{\pi_i\}$, $1 \le i \le n$, разбиения $\pi_{i_1}, \pi_{i_2}, \ldots, \pi_{i_r}$ так, чтобы

$$\bigcap_{j=i_1}^{i_r} \pi_j \le \tau_i.$$

Образуем разбиение C_i:

$$C_i = \bigcap_{j=i_1}^{i_r} \pi_j,\ j \ne i.$$

Принимаем $I_i' = C_i$ и $I_i'' = \eta_i$. Тем самым $I_i = C_i \times \eta_i$. Принимаем $S_i = \pi_i$. Функция переходов компонентного автомата определяется следующим образом:

$$\delta_i : \pi_i \times (C_i \times \eta_i) \to \pi_i.$$

$$\delta_i(\alpha, \beta, \gamma) = \begin{cases} \pi_i(\delta(\alpha \cap \beta, \gamma)), \text{если } \alpha \cap \beta \ne \Phi; \\ \text{не определена, если } \alpha \cap \beta = \Phi. \end{cases}$$

При этом $\delta(\alpha \cap \beta, \gamma) = \{ s' / s' = \delta(t, x),\ t \in \alpha \cap \beta,\ x \in \gamma \}$.

Определим функции $f_i : (\times S_j) \to I_i'$ и $\psi_i : I \to I_i''$, $1 \le i \le n$ следующим образом:

$$f_i(s_1, s_2, \ldots, s_n) = C_i\Big(\bigcap_{j=i_1}^{i_r} s_j\Big);$$
$$\psi_i(x) = \eta_i(x).$$

Выходная функция сети $g : (\times S_j) \times I \to 0$, $1 \le j \le n$, находится в соответствии с выражением

$$g((s_1, s_2, \ldots, s_n), x) = \begin{cases} \lambda\Big(\bigcap_{i=1}^{n} \pi_i, x\Big), & \text{если } \bigcap_{i=1}^{n} \pi_i \ne \Phi; \\ \text{не определена, если } \bigcap_{i=1}^{n} \pi_i = \Phi. \end{cases}$$

В [5] доказано, что построенная таким образом сеть N реализует автомат A.

В дальнейшем считаем, что сеть N – это декомпозиция автомата A.

Пусть $p = x_1 x_2 \ldots x_k$ – последовательность входных букв из алфавита I. Пустую входную последовательность обозначим e, множество всех входных последовательностей конечной длины – через I^*. Через $\bar{\delta}$, $\bar{\lambda}$ и $\bar{g}$ обозначаем обобщенные функции переходов и выходов. Причем $\bar{\delta}(s, p)$ – то состояние, в которое автомат, находящийся в начальный момент времени в состоянии s, переходит при воздействии входной последовательности p. $\bar{\lambda}(s, p)$ – последовательность выходных символов автомата, со-

ответствующая начальному состоянию s и входной последовательности p.

Определение 1. [4] Входную последовательность $p \in I^*$ называем диагностической последовательностью (ДП) для автомата A, если для всех $s, t \in S$ выполняется условие

$$\bar{\lambda}(s, p) = \bar{\lambda}(t, p) \Leftrightarrow s = t.$$

Определение 2. [3] Автомат A называется диагностируемым, если для него существует ДП.

Определение 3. Входную последовательность $p \in I^*$ называем диагностической для компонентного автомата A_i сети N (обозначим ДП$_i$), если для всех $(s_1, s_2, \ldots, s_n), (t_1, t_2, \ldots, t_n) \in X S_i$, $1 \le i \le n$, выполняется условие

$$\bar{g}[(s_1, s_2, \ldots, s_n), p] = \bar{g}[(t_1, t_2, \ldots, t_n), p] \Leftrightarrow$$

$$\Leftrightarrow np_i(s_1, s_2, \ldots, s_n) = np_i(t_1, t_2, \ldots, t_n).$$

Определение 4. Сеть N называем диагностируемой, если для каждого ее компонентного автомата A_i существует ДП$_i$.

Определение 5. [4] Входную последовательность $p \in I^*$ называем установочной последовательностью (УП) автомата A, если для всех $s, t \in S$ выполняется условие

$$\bar{\lambda}(s, p) = \bar{\lambda}(t, p) \Leftrightarrow \bar{\delta}(s, p) = \bar{\delta}(t, p).$$

Определение 6. Входную последовательность $p \in I^*$ называем установочной для компонентного автомата A_i сети N (обозначаем УП$_i$), если для всех $(s_1, s_2, \ldots, s_n), (t_1, t_2, \ldots, t_n) \in X S_i$, $1 \le i \le n$, выполняется условие

$$\bar{g}[(s_1, s_2, \ldots, s_n), p] = \bar{g}[(t_1, t_2, \ldots, t_n), p] \Leftrightarrow$$

$$\Leftrightarrow np_i \bar{\delta}_N[(s_1, s_2, \ldots, s_n), p] = np_i \bar{\delta}_N[(t_1, t_2, \ldots, t_n), p].$$

Условия существования ДП$_i$ и УП$_i$ для компонентных автоматов A_i. Поскольку мы предположили, что для проведения контрольных экспериментов с сетью N нам доступны лишь вход и выход сети, то нам необходимо найти способ проведения локальных контрольных экспериментов с A_i, наблюдая за входом и выходом сети N.

Основой для дальнейшего является понятие связанной по входной последовательности p разбиение на множестве состояний S автомата A.

Определение 7. Связанным разбиением по входной последовательности $p \in I^*$ называем разбиение τ_p на множестве S такое, что

$$s \equiv t(\tau_p) \Leftrightarrow \bar{\lambda}(s,p) = \bar{\lambda}(t,p).$$

Следующей теоремой определяем условия существования ДП$_i$.

Теорема I. Входная последовательность $p \in I^*$ является диагностической для A_i, если и только если $\tau_p \leq \pi_i$.

Доказательство

Необходимость. Из определения (7) следует, что последовательность p является диагностической для автомата A с точностью до блоков разбиения τ_p. Следовательно, (опред. I) существует взаимооднозначное соответствие φ между множеством выходных последовательностей $\Lambda = \{\bar{\lambda}(s,p) / s \in S\}$ и множеством блоков разбиения τ_p такое, что

$$\varphi(\bar{\lambda}(s,p)) = \tau_p(s).$$

Пусть $G = \{\bar{g}((s_1,s_2,\ldots,s_n),p)\ (s_1,s_2,\ldots,s_n) \in S_N\}$. Множество G представляет тем самым всевозможные различные выходные последовательности – реакции сети N на входную последовательность p.

Определяем соответствие φ' между множествами G и π_i следующим образом:

$$\varphi'(\bar{g}((s_1,s_2,\ldots,s_n),p)) = \pi_i(np_i(s_1,s_2,\ldots,s_n)),$$

где $\pi_i(np_i(s_1,s_2,\ldots,s_n))$ – блок разбиения π_i, равный i-му компоненту вектора $(s_1,s_2,\ldots,s_n)$.

Ввиду того, что соответствие φ взаимно однозначное и что выполняются условия $\tau_p \leq \pi_i$ и $\bar{g}((s_1,s_2,\ldots,s_n),p) = \bar{\lambda}(s,p)$,

где $s = \bigcap_{i=1}^{n} s_i$, φ' является однозначным. Отсюда следует, что входная последовательность p различает блоки разбиения π_i, являющиеся внутренними состояниями компонентного автомата A_i. Тем самым p является диагностической последовательностью для A_i.

Достаточность. Нетрудно видеть, что в случае, если условие $\tau_p \leq \pi_i$ не выполняется, соответствие φ' является неоднозначным. Отсюда следует, что p не является диагностической последовательностью для A_i.

Следующая теорема определяет необходимые и достаточные условия для существования установочной последовательности для компонентного автомата A_i.

Теорема 2. Входная последовательность $p \in I^*$ является установочной для A_i если и только если

$$\forall B_j \in \tau_p \exists C_k \in \pi_i [\bar{\delta}(B_j, p) \subset C_k],$$

где

$$\bar{\delta}(B_j, p) = \bigcup_{s \in B_j} \bar{\delta}(s, p).$$

Доказательство теоремы аналогично доказательству теоремы I.

Сетевое структурное дерево автомата. Нахождение $ДП_i$ и $УП_i$. Покажем, что $ДП_i$ и $УП_i$ для A_i могут быть найдены на основе т.н. сетевого структурного дерева (ССД) автомата A. ССД - это некоторая модификация дерева преемников [4].

Таблица I

S \ I	a	b	a	b
1	1	4	0	1
2	1	5	0	0
3	5	5	0	0
4	3	4	1	0
5	2	1	1	0

Корнем ССД принимаем разбиение $\tau_e = 1$, а листьями связанные разбиения τ_p, отмеченные поблочно множествами $\bar{\delta}(B_i, p)$. Лист считаем конечным, если $\forall B_i \in \tau_p [\| \delta(B_i, p) \| = 1]$.

ССД для автомата A (табл. I) приведен на рисунке I. При этом множества $\bar{\delta}(B_i, p)$ приведены над соответствующими блоками разбиения τ_p. Отметим, что автомат является диагностируемым, если в ССД найдется $\tau_p = 0$. Рассматриваемый нами автомат A недиагностируемый.

Предположим теперь, что автомат A декомпозирован по полной системе разбиений

$$\pi_1 = \{\overline{1,2};\ \overline{3,4,5}\} \text{ и } \pi_2 = \{\overline{1,5};\ \overline{2,3};\ \overline{4}\}.$$

Сеть N, являющаяся декомпозицией A, приведена на рисунке 2. Поскольку выполняются следующие неравенства $\tau_{aa} < \pi_1$ и $\tau_{bba} < \pi_2$ и последовательности a и ab удовлетворяют условию теоремы 2 соответственно для π_1 и π_2, то $ДП_I = aa$; $ДП_2 =$

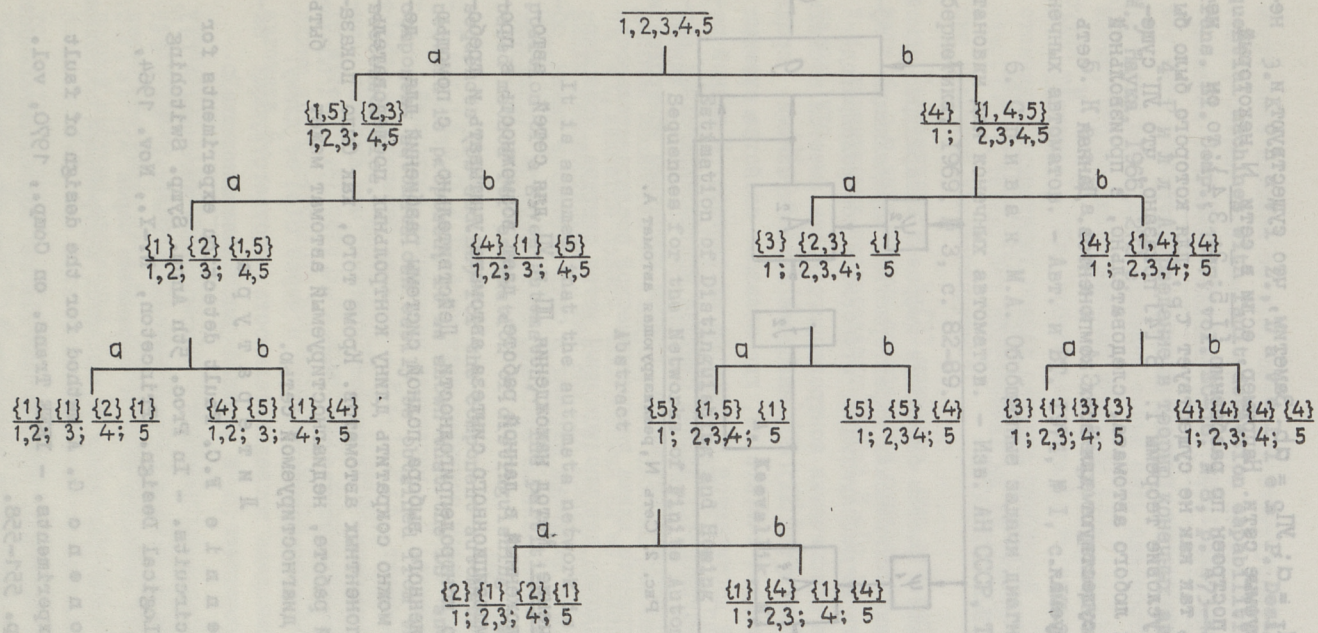


Рис. 1. Сокращенное ССД для автомата А.

= bba; $УП_I$ = a; $УП_2$ = ab. Заметим, что существуют и недиагностируемые сети. Например, если в сети N некоторый компонент построен по разбиению $\{\overline{1,5};\ \overline{2};\ \overline{3,4}\}$; то он не имеет $ДП_i$, так как не существует τ_p, для которого было бы выполнено условие теоремы I. В [4] показано, что УП существует для любого автомата, следовательно, в произвольной сети $УП_i$ существуют для всех компонентов. Данная сеть диагностируемая.

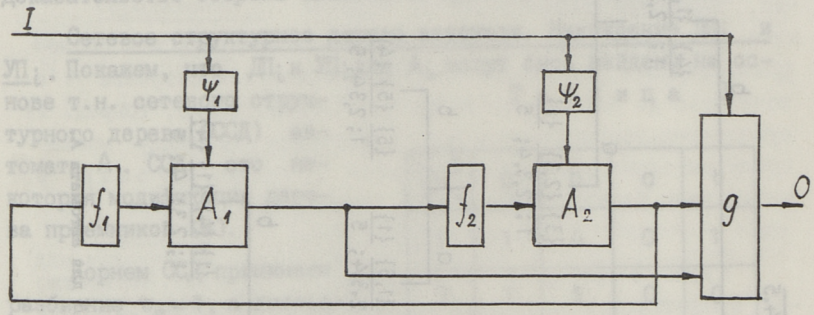


Рис. 2. Сеть N, реализирующая автомат A.

Заключение. Метод нахождения ДП и УП для сетей автоматов, предложенный в данной работе, дает возможность в процессе декомпозиционного синтеза автомата учитывать и требование к его контролепригодности. Действительно, с помощью целенаправленного выбора полной системы разбиений для декомпозиции можно сократить длину контрольных последовательностей компонентных автоматов. Кроме того, как было показано в данной работе, недиагностируемый автомат может быть реализован диагностируемой сетью.

Литература

1. Hennie F.C. Fault detection experiments for sequential circuits. - In Proc. 5th Annu. Symp. Switching Theory and Logical Design. Princeton, N.-Y., Nov. 1964, p. 95-110.

2. Gonenc G. A method for the design of fault detection experiments. - IEEE Trans. on Comp., 1970, vol. C-19, N 6, p. 551-558.

3. К о h a v i Z., L a v e l l e e P. Design of sequential machines with fault detection capabilities. - IEEE Trans. El. Comp., 1967, vol. EC-16, N 8, p. 473-484.

4. Г и л л А. Введение в теорию конечных автоматов. - М.: Наука, 1966. 272 с.

5. К е э в а л л и к А.Э. Теорема декомпозиции конечных автоматов. - Авт. и ВТ, 1974, № 1, с. 17-24.

6. С п и в а к М.А. Обобщенные задачи диагноза и установки для конечных автоматов. - Изв. АН СССР, Техн. кибернетика, 1969, № 3, с. 82-89.

A. Keevallik, M. Kruus

Estimation of Distinguishing and Homing Sequences for the Networks of Finite Automata

Abstract

It is assumed that the automata network is a decomposition of a given automaton. The partition pair algebra based method for estimation of distinguishing and homing sequences (DS and HS) for the component automata of automata network is proposed. It is shown that the DS and HS for component automata can easily be obtained from the modified successor tree.

№ 626

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 62-507

М. Круус

ДЕКОМПОЗИЦИОННЫЙ МЕТОД СИНТЕЗА КОНТРОЛЕПРИГОДНЫХ ЦИФРОВЫХ АВТОМАТОВ

В связи с ростом сложности цифровой аппаратуры одной из важнейших технических характеристик, определяющих эффективность ее эксплуатации, становится контролепригодность. Для обеспечения контролепригодности имеется несколько различных путей [1, 2], наиболее перспективными среди которых следует считать те, где желаемые свойства схемных реализаций достигаются в процессе их синтеза. В данной работе рассматривается задача синтеза контролепригодных абстрактных цифровых автоматов, причем критерием контролепригодности принята длина диагностической последовательности (ДП). Предложен декомпозиционный метод синтеза, позволяющий разлагать заданный автомат в сеть автоматов таким образом, что компонентные автоматы сети имели бы ограниченные по длине ДП.

Под автоматом понимаем пятерку $A=(I,S,O,\delta,\lambda)$, где I - входной алфавит, S - множество внутренних состояний, O - выходной алфавит, $\delta: S\times I \to S$ - функция переходов и $\lambda: S\times I \to O$ - функция выходов. В дальнейшем считаем, что автомат минимальный и сильносвязанный. Через $p = x_1 x_2 \ldots x_k, x_i \in I$ обозначим входную последовательность, а через $d(p)$ ее длину. $\bar{\lambda}(s,p)$ - выходная последовательность, соответствующая начальному состоянию s и входной последовательности p.

Пусть сеть автоматов N - декомпозиция автомата A [4], построенная по полной системе разбиений $P=\{\pi_i\}$, $1 \leq i \leq l$. Компонентные автоматы сети N обозначим через A_i, $1 \leq i \leq l$. При этом, блоки разбиения π_i являются внутренними состояниями A_i.

В [4] показано, что диагностические последовательности ($ДП_i$) для компонентных автоматов A_i, если они существуют,

могут быть найдены по т.н. сетевому структурному дереву (ССД), опираясь на понятие связанного по входной последовательности разбиения. Покажем здесь, что любой автомат A может быть декомпозирован таким образом, что все компонентные автоматы имеют ДП$_i$. Называем такую сеть диагностируемой. Отметим, что не любой автомат имеет ДП [3].

Теорема I. Для любого автомата A с n внутренними состояниями существует полная система разбиений $P=\{\pi_i\}$, $1\le i\le n-1$, такая, что построенная на ее основе декомпозиция является диагностируемой.

Доказательство. В [3] показано, что для любого автомата A с n внутренними состояниями существует множество т.н. характерных последовательностей $R=\{p_1,p_2,\dots,p_\ell\}$, $\ell\le n-1$, такая, что

$$(\forall p_j\in R\,[\bar\lambda(s,p_j)=\bar\lambda(t,p_j)])\Leftrightarrow s=t,$$

где $s,t\in S$.

Находим для каждой последовательности p_j связанное разбиение τ_{p_j} [4]. Из определения связанного разбиения и определения множества R следует, что система разбиений $\{\tau_{p_j}\}$, $1\le j\le\ell$, является полной. Построенная на ее основе декомпозиция диагностируемая, так как для компонентного автомата, построенного по τ_{p_j}, последовательность p_j является диагностической. Теорема доказана.

Следующая теорема устанавливает минимальную возможную длину диагностической последовательности для компонентного автомата A_i.

Теорема 2. Максимальная по длине ДП$_i$ для компонентного автомата A_i сети N не может быть короче максимальной по длине диагностической последовательности для двух состояний автомата A.

Доказательство. Предположим, что у автомата A пара состояний (s,t) различима последовательностью $p_{s,t}$, $d(p_{s,t})=q$. Предположим, также, что задана декомпозиция автомата A в диагностируемую сеть N, причем декомпозиция построена по полной системе разбиений $\{\pi_i\}$, $1\le i\le\ell$ и $\max_i(d(\text{ДП}_i))<q$. Известно [4], что если последовательность p является ДП$_i$, то соответствующее связанное разбиение $\tau_p\le\pi_i$. Следовательно, если все ДП$_i$ короче q, то $s\equiv t(\tau_p)$ (s и t содержатся

в одном блоке разбиения τ_p) для всех связанных разбиений, а также $\delta \equiv t(\pi_i)$, $1 \le i \le l$. Тем самым показано, что система разбиений не может быть полной. Это противоречие доказывает теорему.

В [4] было введено понятие ССД. Дополним условия прерывания ветвей ССД. А именно, прерываем все ветви ССД на к-том уровне, если $\prod_{d(p)\le k} \tau_p = 0$. Полученное дерево называем сокращенным ССД. Сокращенное ССД для автомата А (табл. I) приведено на рис. I. Из теоремы 2 следует, что количество

Таблица I.

s \ I	a	b	a	b
1	1	4	0	1
2	1	5	0	0
3	5	5	0	0
4	3	4	1	0
5	2	1	1	0

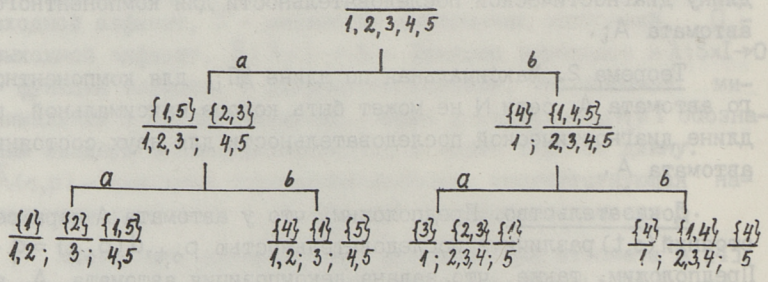


Рис. 1. ССД для автомата А.

уровней в сокращенном ССД определено длиной максимальной диагностической последовательности $p_{\delta,t}$. Действительно, так

как $\prod_{d(p)\leq\kappa} \tau_p = 0$, то для всех пар состояний (s,t) должно существовать τ_p такое, что $s \not\equiv t(\tau_p)$, а следовательно, p различает состояния s и t. Очевидно, что на основе сокращенного ССД можно выбирать полную систему разбиений для построения диагностируемой сети, причем $\max_i(d(\Delta\Pi_i)) = \kappa$, где κ - количество уровней сокращенного ССД. Называем такую декомпозицию κ-диагностируемой.

Предложим метод для выбора полной системы разбиений для построения κ-диагностируемой декомпозиции. Построим сокращенное ССД и находим на ее основе множество разбиений $T=\{\tau_{p_i}\}$. Из множества T выбираем минимальное подмножество разбиений T' такое, что $\prod_i \tau_{p_i}=0, \tau_{p_i}\in T'$. Составим полную систему разбиений $\{\pi_i\}$ такую, что $\forall \pi_i \exists \tau_{p_i}\in T'[\pi_i \geq \tau_{p_i}]$. Построенная на основе этой системы декомпозиция κ-диагностируемая, причем $\Delta\Pi_i = p_i$. Для автомата A (табл. I) множество T состоит из 4 разбиений: $\tau_a=\{\overline{1,2,3};\ \overline{4,5}\}$, $\tau_b=\{\overline{1};\ \overline{2,3,4,5}\}$, $\tau_{aa}=\tau_{ab}=\{\overline{1,2};\ \overline{3};\ \overline{4,5}\}$, $\tau_{ba}=\tau_{bb}=\{\overline{1};\ \overline{2,3,4};\ \overline{5}\}$. В подмножестве T' минимально два разбиения. Выбираем $T'=\{\tau_{aa},\tau_{bb}\}$. На основе T' составим полную систему разбиений $\{\pi_1,\pi_2\}$, где

$$\pi_1=\{\overline{1,2};\ \overline{3};\ \overline{4,5}\}=\tau_{aa};$$

$$\pi_2=\{\overline{1,5};\ \overline{2,3,4}\}>\tau_{bb}.$$

Построенная по этой системе декомпозиция 2 - диагностируемая, причем $\Delta\Pi_I = aa$; $\Delta\Pi_2 = bb$.

Отметим, что условие полноты системы разбиений для декомпозиции можно выполнить не только разбиениями с κ-го уровня сокращенного ССД, но и с разбиениями с предыдущих уровней. Тем самым можно получить реализацию автомата A в виде сети, где κ - максимальная длина для $\Delta\Pi_i$.

Литература

1. Fujwara H., Kinochita K. Design of diagnosable sequential machines utilizing extra outputs. - IEEE Trans. on Comp., vol. C-23, N 2, p. 138-145.

2. Murakami S., Kinoshita K., Ozaki H. Sequential machines capable of fault diagnosis. - IEEE Trans. on Comp., 1970, vol. C-19, N 11, p. 1079-1085.

3. Hennie F.C. Fault defection experiments for sequential circuits. - Proc. 5th Annu. Symp. on Switching Theory and Logical Design. Princeton, N.-Y., Nov., 1964, p. 95-110.

4. Кеэвалдик А.Э., Круус М.Э. Метод нахождения диагностических и установочных последовательностей для сетей автоматов. См. наст. сб., с. 45.

M. Kruus

Decomposition Method for Easily Diagnosable Sequential Machine Design

Abstract

This paper deals with the design of easily diagnosable sequential machines. The length of the distinguishing sequence is regarded as the criterion of diagnosability. It is shown that for any arbitrary machine there exists a diagnosable network, in which all the component machines have the distinguishing sequences. A decomposition method for k-diagnosable network design is proposed.

№ 626

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.32

Р.Р. Убар

МЕТОДЫ ТЕСТОВОГО ДИАГНОСТИРОВАНИЯ ДИСКРЕТНЫХ СИСТЕМ

Использование прогрессивных технологий в микроэлектронике, переход на новую элементную базу и использование новых архитектурных решений при проектировании изделий вычислительной техники вызвали моральное старение диагностических средств, ориентированных на вентильный уровень представления дискретных объектов (ДО) и потребовали развития теории технической диагностики и разработки новых методов и средств, пригодных для автоматизации диагностирования сегодняшних дискретных систем (ДС). С другой стороны, необходимость многоуровневого представления ДС вызвала появление целой совокупности различных математических моделей и методов, что в конечном итоге затрудняет создание систем автоматизированного проектирования диагностического обеспечения и увеличивает затраты на их создание. Отсюда следует актуальность разработки единого и комплексного подхода к диагностике ДО с целью создания универсальных средств диагностирования, пригодных для широкого класса ДС.

Ниже излагаются в сжатой форме результаты, полученные в течение последних 10-ти лет на кафедре ЭВМ ТПИ в области технической диагностики дискретных объектов, направленные на решение вышепоставленной задачи [1, 2].

В качестве математического аппарата для создания модели диагностического эксперимента и исследования его свойств был выбран аппарат булева дифференциального исчисления. По сравнению с другими возможными способами задания ДО он обеспечивает более компактное, а также единое представление основных задач диагностирования, таких как синтез и анализ тестов и дешифрация результатов диагностических экспериментов.

Рассмотрим ДС в виде множества булевых функций F на множестве переменных Z, причем любая функция $z_k = f_k(Z_k)$, где $z_k \in Z$, $Z_k \subset Z$ и $f_k \in F$, соответствует некоторому компоненту ДС (или ее подсхеме). В [3, 4] показана возможность применения булева полного дифференциала

$$dz_k = df_k(Z_k \oplus dZ_k) \qquad (I)$$

в качестве основы при создании математической модели диагностического эксперимента над ДС. Так решение уравнения (I) при заданном Z_k можно рассматривать как прямую, а при заданном dZ_k – как обратную задачу диагностирования ДС. Проведена классификация различных задач тестовой диагностики, таких как построение тестов, моделирование и анализ тестов, дешифрация результатов диагностических экспериментов и т.д. как частных случаев решения уравнения диагноза (I). Так, в частном случае, при одиночных неисправностях уравнение диагноза упрощается и решение его в обоих случаях сводится к вычислению частных булевых производных [I, 2].

Введение понятия уравнения диагноза в виде (I) позволило расширить класс формально рассматриваемых неисправностей не только при решении задач построения тестов, но и при дешифрации результатов диагностических экспериментов. Обозначив через дифференциал dz_k ложное изменение значения переменной z_k, получим для описания неисправных ситуаций в диагностическом эксперименте импликации вида

$$dz_k \rightarrow \bigvee_i r_{ki} W_{ki}, \quad i : r_{ki} \in R_k, \qquad (2)$$

где R_k – множество неисправностей, местонахождение которых определено переменной z_k ($r_{ki} = 0$ соответствует отсутствию i-й неисправности; а $r_{ki} = I$ – присутствию ее),

W_{ki} – дополнительное логическое условие, необходимое для локальной активизации неисправности r_{ki} в точке z_k.

При выполнении определенных ограничений импликацию (2) удается привести к равенству, что позволяет в уравнении диагноза (I) учитывать кроме константных неисправностей также произвольные логически описываемые неисправности широкого класса, такие как короткие замыкания, перепутывания связей, функциональные неисправности компонентов и т.д. [5, 6]. Это

позволило разработать общие формальные методы синтеза и анализа тестов для широкого класса неисправностей. Предложенный дифференциальный подход дает возможность формализовать также обработку неисправностей, приводящих к увеличению числа состояний в объекте [7]. Введение класса функциональных неисправностей компонентов ДС обеспечивает снижение размерности ДО, а также формальный переход от уровня к уровню при иерархическом подходе к проектированию тестов [5, 6].

Аппарат булева дифференциального исчисления использовался традиционно для решения обратной задачи диагностирования для присвоения тестов. В работах [5, 6] на основе уравнения диагноза (I) впервые поставлена и решена прямая задача диагностирования - задача дешифрации результатов диагностического эксперимента для расширенного класса неисправностей. В работе [8] рассматривалась прямая задача анализа тестов для частного случая одиночных неисправностей.

Предложенный аналитический подход к описанию диагностических экспериментов в виде булева полного дифференциала оказывается полезным с точки зрения исследования взаимосвязей различных задач диагностирования. Конкретные же операции и алгоритмы на базе этого аппарата теряют эффективность при росте размерности ДО. С целью разработки эффективных алгоритмов диагностирования, общих для широкого класса объектов, был разработан аппарат альтернативных графов (АГ) [9--II].

Основным существенным свойством рассматриваемой АГ-модели является возможность непосредственной и наглядной интерпретации в ней физической сущности уравнения полного булева дифференциала как модели диагностического эксперимента. При этом многие задачи решаются на АГ-модели, благодаря ее однородности, существенно проще, чем средствами аналитического аппарата.

Рассмотрим АГ-модель компонента ДС с функцией произвольного вида $z_K = f_K(Z_K)$ в виде ориентированного графа $G_K = (M_K, \Gamma_K, Z_K)$, где M_K - множество вершин, Γ_K - отображение M_K в M_K, а Z_K - множество переменных (аргументов функции f_K). Переменные $z \in Z_K$ задаются своими областями определения $A(z) = \{0,1,\ldots\}$. Каждой вершине $m \in M_K$ приписана некоторая весовая функция

$$e(m) = f_{K,m}(Z_{K,m}), \qquad (3)$$

где $Z_{K,m} \subseteq Z_K$.

Определено отображение $\Gamma(m,e(m))$, по которому каждому значению $e(m)$ однозначно или соответствует некоторый последователь m, $\Gamma(m,e(m)) \in \Gamma(m)$ или $\Gamma(m,e(m)) = \Phi$. Частным случаем АГ являются простые двоичные АГ, гле функции (3) вырождены в двоичные переменные $z(m) \in Z_K$ [9]. Частным случаем последних являются бинарные деревья решений (БДР), которые в [I2] были предложены также для решения задач построения тестов ДО.

Известно, что асимптотической оценкой максимального числа вершин в БДР для n-аргументной булевой функции является $2^n/n$ [I3]. Но можно показать, что для наиболее типичных для практики компонентов ДС сложность АГ-модели возрастает линейно с ростом n. В табл. I приведены примеры сравнения сложностей для типичных компонентов ДО при представлении их АГ-моделью и в виде дизъюнктивной нормальной формы (ДНФ).

Таблица I

№	Дискретный объект	АГ-модель (число вершин)	ДНФ (число букв)
I	n-входовые логические элементы /И, ИЛИ, И-НЕ, ИЛИ-НЕ/	n	n
2	n-входовая схема нечётности	$2n-1$	$n2^{n-1}$
3	мажоритарный элемент " m из n "	$m(n-m+1)$	mC_n^m
4	n-разрядный сумматор	$5n$	$12n$
5	n-входовый мультиплексор	$2n-1$	$(\log_2 n+1)n$
6	n-входовый демультиплексор	$2(n-1)+1$	$(\log_2 n+1)n$
7	n-входовый дешифратор	$2n-1$	$n2^n$

Сжатость АГ-модели следует также из возможности совмещения в одной графе как прямой, так и обратной функций

ДО без их отдельного представления, как это традиционно делается при использовании нормальных форм. Кроме сжатости, положительным качеством АГ-модели является ее регулярность, позволяющая упростить алгоритмы синтеза и анализа тестов и увеличить их эффективность по сравнению с применением традиционных моделей ДО в виде скобочной формы (СФ) булевой функции или структурно-аналитического описания ДО.

Преимуществом АГ-модели, предложенной в [9], по сравнению с БДР [12] является возможность отображения в модели структуры ДО, а следовательно, и структурных неисправностей. В [9] предложен способ синтеза АГ-модели так, чтобы каждая вершина в АГ представляла некоторый путь (или часть пути) в исходной схеме ДО. Таким образом, традиционная задача проверки путей в ДО на языке АГ может быть приведена к задаче проверки вершин. Классификация АГ на структурные и функциональные и исследование их свойств позволили наметить пути их эффективного использования: структурные АГ целесообразны при построении тестов для компонентов (подсхем, модулей) ДО, когда размерность объекта относительно мала, а функциональные АГ, обеспечивающие произвольное сжатие, целесообразны при построении тестов на уровне всего ДО с целью активизации путей через отдельные компоненты [1, 2].

Существование непосредственной связи между аппаратами булева дифференциального исчисления и АГ обеспечивает возможность комплексного решения на АГ как прямых, так и обратных задач диагностирования ДО, расширяя в то же время рассматриваемый класс объектов, охватывая не только нерегулярные ДО средней размерности, которые описываются традиционно аппаратом булевых функций, а также ДС высокой размерности, таких, как микропроцессорные БИС и системы. Рассмотрим вначале выполнение различных процедур тестового диагностирования ДО на простых двоичных АГ.

Моделирование тестового набора T сводится к движению на АГ такому, чтобы направление выхода из каждой вершины $m \in M_K$ однозначно определялось значением переменной $z(m) \in Z_K$. В результате моделирования набора T в графе активизируется некоторый путь $L(T) \subseteq M_K$ с конечной вершиной m', $\Gamma(m', z(m')) = \Phi$. Значение функции f_K, определяемое в результате моделирования, будет равно значению перемен-

ной $z(m')$. Число шагов при моделировании в общем случае меньше, чем $|M_K|$. Возможно и параллельное моделирование нескольких наборов с числом шагов, равным $|M_K|$.

Анализ полноты тестового набора T сводится к вычислению значений частных булевых производных $\partial z_K/\partial z(m)$, где $z(m) \in L(T)$. На АГ-модели это эквивалентно проведению повторного моделирования набора T для всех вершин $m \in L(T)$ при изменении направления выхода из вершины. Пусть $m''(m)$ - конечная вершина, достигнутая при повторном моделировании набора из вершины m. Тогда имеем

$$z(m''(m)) \neq z(m') \Leftrightarrow \frac{\partial z_K}{\partial z(m')} = 1. \qquad (4)$$

Анализ полноты может быть проведен также параллельно для нескольких тестовых наборов с числом шагов, равным $|M_K|$.

Многозначное моделирование тестовых наборов с целью анализа их корректности в отличие от известных методов при использовании АГ может быть проведено на той же модели, которая используется при двоичном моделировании. Моделирование основывается на вычислении максимума булевой производной

$$\delta(m) = \max_{z(m_i)} \{\partial f_K/\partial z(m)\}, \qquad (5)$$

где $m, m_i \in M_K$ и $m \neq m_i$.

Функция (5) представляет собой обобщение булевой производной для случая многозначных сигналов. При $\delta(m)=0$ для всех $m \in M_K$ значение функции f_K на данном наборе будет статическим. В противном случае конкретное динамическое значение функции f_K при заданном алфавите сигналов может быть определено по соответствующим решающим таблицам [I4]. Число шагов при моделировании меньше или равно $|M_K|$. Преимуществом метода по сравнению с известными является его универсальность как относительно элементной базы, так и относительно выбранного алфавита моделирования.

Известные подходы к моделированию ДО, базирующиеся на алгебре логики, не применимы в случае обработки длинных последовательностей, состоящих из сотен тысяч и миллионов тактов. Такие последовательности типичны для ДС, содержащих счетные структуры. В [I5] разработан новый подход к моделированию "длинных" последовательностей, где логические

методы анализа потенциальных сигналов сочетаются с арифметической обработкой числа фронтов в импульсных пачках. Разработанный в [15] метод является дальнейшим развитием идеи многозначного моделирования на АГ, предложенного в [14].

Генерирование тестового набора для переменной $z(m)$ при некоторой вершине $m \in M_K$ сводится к активизации трех путей: пути из начальной вершины АГ к вершине m и двух путей, выходящих из m в разных направлениях и входящих в разные висячие вершины m' и m'' так, чтобы, согласно (4), выполнялось $z(m') \neq z(m'')$. Соответствующие алгоритмы генерирования тестов предложены в [10, 11]. При этом показано, что на АГ-модели задача активизации многомерных путей сводится к задаче активизации одномерных путей в АГ, что существенно сокращает перебор вариантов при синтезе тестов. По сравнению с известными аналитическими методами на АГ-модели упрощаются задачи синтеза парных и групповых тестов, предназначенных для обнаружения кратных неисправностей [16]. Регулярность АГ-модели, возможность компактного описания приоритетности сигналов и режимов, а также запрещенности определенных комбинаций сигналов позволяет упростить алгоритмы перебора и увеличить производительность при построении тестов для последовательностных схем.

Дешифрация результатов диагностических экспериментов основывается на моделировании отказавшего тестового набора T и определении на АГ пути $L(T)$. При обнаружении ложного сигнала на выходе компонента f_K потенциальными источниками ложного сигнала являются входы компонента $z(m) \in Z_K$, соответствующие вершинам $m \in L(T)$. Определен ряд свойств АГ-модели, позволяющих минимизировать множество $L' \subseteq L(T)$ проверяемых точек [17, 18]. На основе этих свойств в [17] разработаны принципы управления процессами поиска дефектов в ДО, которые не требуют хранения в памяти тестера больших массивов эталонной информации и генерирование которой происходит оперативно при обнаружении ложного сигнала путем моделирования непрошедшего теста на АГ-модели. Предложены алгоритмы выработки оптимизированных стратегий для управления поиском дефектов. Применение АГ-модели позволяет увеличить точность локализации дефектов, а также уменьшить трудоемкость операций дешифрации по сравнению с аналитическими методами.

Изложенные выше теоретические результаты реализованы в системе автоматизированного проектирования тестов для ДО (комплекс программ на ЕС ЭВМ) и в диалоговой системе диагностирования ДО (тестер, управляемый от СМ-4), разработанных совместно в СКБ Вычислительной техники Института кибернетики АН ЭССР и на кафедре ЭВМ ТПИ. Обе системы основываются на сквозном представлении информации о ДО в виде АГ, обеспечивая, таким образом, комплексный подход к автоматизации как проектирования тестов, так и управления диагностическими экспериментами над ДО. Последнее обстоятельство существенно снижает затраты на эксплуатацию этих систем, в том числе затраты на ручную подготовку исходной информации и составления архива элементной базы.

Система автоматизированного проектирования тестов, основные принципы работы которой описаны в [19, 20], предназначена для автоматизации построения и анализа тестов для блоков дискретных устройств, дискретных систем, построенных на базе ИС, СИС и БИС, различного рода матричных БИС, а также для микропроцессорных БИС и систем. Для повышения производительности системы при генерировании тестов имеются некоторые средства для интерактивного ввода частично определенных тестовых наборов, микропрограмм или временных диаграмм. Некоторые примерные данные эксплуатации системы приведены в табл. 2.

Система диагностирования ДО, принципы работы которой рассмотрены в [17, 18], представляет собой программно-управляемый стенд, состоящий из ЭВМ СМ-4, тестера и специального программного обеспечения, и предназначена для автоматического контроля ТЭЗов ЭВМ и поиска в них дефектов в диалоговом режиме при помощи ручного пробника. В системе не требуется хранение заранее вычисленных таблиц эталонных реакций, нужных для работы с пробником. Необходимая минимальная эталонная информация генерируется в момент обнаружения дефекта путем селективного моделирования соответствующей части отказавшего теста. Одновременно строится оптимизированная стратегия поиска дефектов путем минимизации и упорядочения подозреваемых контрольных точек в виде диагностического дерева. Статистическими экспериментами показано, что применение разработанных в [17, 18]методов отсеивания бесперспективных вариантов поиска обеспечивает в

Таблица 2

№	Проверяемый объект	Характеристики объекта				Характеристики теста			
		Количество				Кол-во неиспр.	Время (мин.)	Длина (тактов)	Полнота (%)
		Входов	Выходов	Микро-схем	Элементов памяти				
I	ЕС 2060/0233	33	56	I4	-	274	I.05	60	I00
2	ЕС 2060/033I	84	4	30	-	332	0.22	29	I00
3	ЕС 2060/038I	39	55	3I	3	49I	3.06	59	I00
4	ЕС 7022/0020	I3	2I	I0	I8	I76	0.27	60	I00
5	ЕС 7022/0023	I7	I8	20	-	346	2.50	82	I00
6	ЕС 7022/005I	I8	I5	I3	I0	I96	I.35	70	99
7	Спец. КУ	80	4	37	-	522	3.00	37	I00
8	Спец. ПУ	2I	7	I2	8	I84	2.00	49	90
9	Спец. ПУ	29	4	32	48	472	I4.00	202	92
I0	Устр-во контроля	52	2	67	-	I079	7.00	92	I00
II.	Арифм. расшир.	45	I4	I8	-	35I	2.07	30	95
I2	Операц. автомат	I0	I8	43	77	760	28.00	235	88
I3	Сдвиг. регистр	26	3I	32	30	653	5.00	6I	94
I4	Процессор ЭВМ	52	30	I05	68	2I40	(45.00) 240.00	358	(50) 85
I5	Устройство упр-ния	I0	I	I6	3	78	(9.23) 0.58	40	9I
I6	Устройство упр-ния	44	I6	30	49	654	22.I2	I59	8I
I7	К580 ИК55	38	32	I	32	I300	25.59	227	82
I8	К58С ИК80 (упр. часть)			I		I034	I6.00	58I	92

среднем 2-3-кратное сокращение ручной работы с пробником. Резкое увеличение производительности тестера при поиске дефектов в случае, когда тест содержит длинные импульсные последовательности, обеспечивают программные средства для аналитической декомпозиции импульсных пачек и вычисления эталонных сигналов для работы с пробником в терминах числа фронтов в импульсной пачке или в ее частях [15].

Рассмотренные выше методы и процедуры диагностирования легко распространяются на общий случай АГ, предназначенный для единого представления широкого диапазона ДО, начиная с нерегулярных ДО, заданных на уровне транзисторных или логических схем, и кончая микропроцессорными системами, задаваемыми микропрограммами или системами команд. Исследованию обобщенных АГ и разработке методов построения тестов, основанных на общем случае АГ, посвящены работы [21-23]. Установлена связь между АГ и известными моделями ДО, такими как булевы функции, топологические модели транзисторных схем БИС, граф-схемы алгоритмов, модели регистровых передач, изложены преимущества АГ по сравнению с перечисленными моделями и показана возможность постановки ширского класса задач технической диагностики ДО, решаемых при помощи перечисленных моделей, на единой основе АГ. Непосредственно из однородности АГ-модели вытекают не только простые алгоритмы обработки модели, но и простые соотношения между тестами и неисправностями.

Результаты работ [21-23] реализованы при разработке системы диагностирования микропроцессорных БИС и систем, состоящей из микроЭВМ СМ-1800, тестера и специального программного обеспечения, и предназначенной для алгоритмического генерирования тестпрограмм на основе структурированного представления тестовой информации в виде программ, массивов вариации программ и массивов данных (операндов). В тестере реализован принцип циклической модификации основной тест-программы путем варьирования определенных ее частей и использования разных операндов. Функциональные возможности тестера способствуют реализации групповых тестов [16, 23], имеющих повышенную чувствительность к кратным неисправностям. Построение тестовой информации происходит на основе АГ-модели проверяемого объекта. Благодаря структурированному представлению тестовой информации резко

сокращается необходимый объем памяти для хранения теста, а также время проверки объекта за счет минимизации объема вводимых данных.

Разработанная концепция диагностической модели дискретных систем в виде альтернативных графов, рассматриваемых, с одной стороны, как язык и программный способ описания объекта, и, с другой стороны, как сжатое представление теста, позволяет объединять возможность сжатого и регулярного представления функций объекта с возможностями однозначной интерпретации его структуры и структурных неисправностей и, таким образом, способствует созданию универсальной системы диагностирования широкого класса ДО на разных уровнях иерархии их описания.

Литература

I. Убар Р.Р. Тестовая диагностика цифровых устройств. I и II части. Таллин, ТПИ, 1981. 226 с.

2. Селезнев А.В., Добрица Б.Т., Убар Р.Р. Проектирование автоматизированных систем контроля бортового оборудования летательных аппаратов. - М.: Машиностроение, 1983. 224 с.

3. Убар Р.Р. Об общей постановке задач тестовой диагностики цифровых схем. - Тр. Таллинск. политехн. ин-та, 1976, № 409, с. 69-73.

4. Ubar R.R. Multiple fault analysis in logical circuits. - Proceedings of IFAC Symposium on Discrete Systems. Dresden, 1977, Band 4, p. 48-57.

5. Ubar R.R. Fehlerbestimmung in kombinatorischen Schaltungen durch Lösung der Boole' schen Differentialgleichungen. - Nachrichtentechnik/Elektronik, 1978, H. 8, S. 330-334.

6. Убар Р.Р. Выделение подозреваемых неисправностей в комбинационных схемах методом решения булевых дифференциальных уравнений. - Автоматика и телемеханика, 1979, № II, с. 170-183.

7. Убар Р.Р. Описание неисправностей цифровых устройств. - Тр. Таллинск. политехн. ин-та, 1980, № 497, с. 3-9.

8. У б а р Р.Р. Анализ диагностических тестов для комбинационных цифровых схем методом обратного прослеживания неисправностей. - Автоматика и телемеханика, 1977, № 8, с. 168-176.

9. У б а р Р.Р. Генерирование тестов для цифровых схем при помощи модели альтернативных графов. - Тр. Таллинск. политехн. ин-та, 1976, № 409, с. 75-81.

10. П л а к к М.П., У б а р Р.Р. Построение тестов цифровых схем при помощи модели альтернативных графов. - Автоматика и телемеханика, 1980, № 5, с. 152-163

11. U b a r R.R. Beschreibung digitaler Einrichtungen mit alternativen Graphen für die Fehlerdiagnose. - Nachrichtentechnik/Elektronik, 1980, 30, H. 3, S. 96-102.

12. A k e r s S.B. Binary Decision Diagrams. - IEEE Trans. on Computer, June, 1978, vol. C-27, p. 509-516.

13. К у з ь м и н В.А. Оценка сложности реализации функций алгебры логики простейшими видами бинарных программ. - В кн.: Методы дискретного анализа и теории кодов и схем. Вып. 29. Новосибирск. Ин-т математики СО АН СССР, 1976, с. 11-39.

14. В о о л а й н е А.А., П а л л ь М.А., У б а р Р.Р. Обобщенный подход к многозначному моделированию цифровых схем на модели альтернативных графов. - Тр. Таллинск. политехн. ин-та, 1982, № 530, с. 23-38.

15. У б а р Р.Р., Э в а р т с о н Т.А. О моделировании длинных входных последовательностей в дискретных устройствах, содержащих счетные структуры. - Тр. Таллинск. политехн. ин-та, 1985, № 601, с. 61-74.

16. У б а р Р.Р. Построение полных контролирующих тестов для комбинационных схем. - Изв. АН ЭССР, том 31, Физика/Математика, 1982, № 4, с. 418-427.

17. L o h u a r u T.V., V i i l u p A.A., U b a r R.R. Minicomputer software for fault location control in digital-circuits. - Preprints of IFAC/IFIP 2nd Int. Symp. on SOCOCO. Prague, 1979, vol. 1, p. XIV.

18. T h o m ä E., U b a r R.R. Optimierte Steuerung

der Fehlersuche auf digitalen Leiterplatten. - 27. Int. Wiss. Koll. TH Ilmenau, 1982, H. 3, S. 65-68.

I9. Вооласйне А.А., Йыги А.Р., Паллъ М.А. Проектирование контрольных экспериментов в автоматизированной системе контроля цифровых автоматов "НАКС". - Тр. Таллинск. политехн. ин-та, 1982, № 530, с. 3-21.

20. Лохуару Т.В., Паллъ М.А., Убар Р.Р. Автоматический синтез тестов для диагностики цифровых устройств. - Изв. АН ЭССР, том 32, Физика/Математика, 1983, № I, с. 84-94.

21. Ubar R.R. Vektorielle alternative Graphen und Fehlerdiagnose für digitale Systeme. - Nachrichtentechnik/ Elektronik, 1981, 31, H. 1, S. 25-29.

22. Ubar R.R. Test pattern generation for digital systems on the vector alternative graph model. - Digest of Papers of 13th Annual Int. Symp. on Fault Tolerant Computing FTCS' 13. Milano, 1983, p. 374-377.

23. Убар Р.Р. Универсальный подход к автоматизации проектирования тестов для широкого класса дискретных объектов. См. наст. сб.,с. 75.

R. Ubar

Methoden zum Testen von digitalen Systemen

Zusammenfassung

Es werden die Ergebnisse, die man am Lehrstuhl für Rechenmaschinen am Tallinner Polytechnischen Institut in den letzten 10 Jahren auf dem Gebiet der technischen Diagnose digitaler Objekte bekommen hat, dargestellt.Man betrachtet ein mathematisches Modell für Diagnoseexperimente in der Form einer Diagnosegleichung, die auf dem Apparat von Boole'schen Differentialkalkül beruht. Die Konzeption des Diagnosemodells für digitale Systeme in der Form von alternativen Graphen, die als Basis für ein universales Diagnosesystem dient, wird dargestellt.

№ 626
TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА
УДК 681.32

Р.Р. Убар

УНИВЕРСАЛЬНЫЙ ПОДХОД К АВТОМАТИЗАЦИИ ПРОЕКТИРОВАНИЯ ТЕСТОВ ДЛЯ ШИРОКОГО КЛАССА ДИСКРЕТНЫХ ОБЪЕКТОВ

Одной из тенденций в производстве цифровых схем и систем является все большее применение элементов высокой интеграции и в связи с этим резкое усложнение функций, реализуемых устройствами. В результате этого увеличиваются трудности, связанные с диагностированием неисправностей этих объектов. Высокая степень интеграции и ограниченное количество контрольных точек в устройствах, построенных на микропроцессорных наборах БИС, делает невозможным применение регулярных методов генерирования тестов, основанных на вентильном представлении контролируемых объектов.

В данной статье рассматривается дальнейшее развитие модели альтернативных графов (АГ), разработанной в [1-4] с целью формализации процесса синтеза тестов для широкого класса дискретных объектов, включая микропроцессорные БИС и системы. В основе предлагаемой обобщенной модели АГ лежит возможность работы не только на множестве двоичных сигналов, но и на множестве групп сигналов, рассматриваемых как единые целые и разнесенных как в "пространстве", так и во "времени". Такими группами сигналов могут служить, например, параллельные и последовательные коды (данные, адреса, команды, состояния микропрограммных автоматов), импульсные пачки и т.д. Модель АГ введена для описания функций компонентов рассматриваемого объекта, получаемых в результате структурной или функциональной декомпозиции объекта. Такими компонентами могут служить, например, произвольные функционально законченные части объекта (если он задан в виде структурной или принципиальной схемы), программно- или микропрограммно-доступные элементы (если объект задан в виде программы, микропрограмм или системы команд).

I. Описание модели альтернативных графов

Рассмотрим объект диагностирования (ОД) в виде

$$ОД = \{Z, F, P\},$$

где $Z = X \cup S \cup Y$ множество переменных (X,S,Y - соответственно подмножества входных, внутренних и выходных переменных), F - множество функций вида $z_i = f_i(Z)$, а P - множество предикатов $z_j = p_j(Z)$, где z_i, $z_j \in S \cup Y$. Переменные $z \in Z$ рассмотрим в общем случае в виде n - разрядных слов, $n = 1,2 \ldots$, принимающих значения из области задания $z_i \in V(z_i) \subseteq \subseteq \{0,1,\ldots,2^n-1\}$. Для логических сигналов имеем $n = 1$.

Переменным $z_i \in S \cup Y$ и следовательно, соответствующим функциям $f_i \in F$ (или предикатам $p_j \in P$) сопоставлены альтернативные графы

$$G_i = \{M_i, \Gamma_i\},$$

где M_i - множество вершин, а Γ_i - отображенные M_i в M_i.

Введем разбиение $M_i = M_i^B \cup M_i^T$, где $M_i^B = \{m \mid \Gamma_i(m) \neq \Phi\}$ множество внутренних вершин, а $M_i^T = \{m \mid \Gamma_i(m) = \Phi\}$ множество терминальных вершин АГ. Вершинам $m \in M_i$ сопоставлены весы $e(m)$, которыми могут быть переменные $z_i \in X \cup S$, предикаты $p(Z)$ или функции $f(Z)$. Терминальные вершины могут быть взвешены также константами. Весы $e(m)$ принимают значения из их областей определения $V(e(m))$. Каждому значению $e(m)$ соответствует определенный последователь вершины $m_i = \Gamma_i(m, e(m)) \in \Gamma_i(m)$. Последователи для различных значений $e(m)$ могут совпадать, следовательно, в общем случае $|\Gamma_i(m)| \leq |V(e(m))|$.

Процесс вычисления значения переменной z_i, представленной графом G_i, может быть трактован как процесс движения по вершинам графа G_i с начальной вершины до некоторой конечной вершины $m^T \in M_i^T$. Искомое значение переменной z_i определяется значением веса в достигнутой вершине m^T, $z_i = e(m^T)$. Итак, каждому пути $L \subseteq M_i$ на графе соответствует некоторая совокупность значений переменных $z \in L \subseteq Z$, встречающихся в весах при вершинах $m \in L$ на этом пути.

В качестве частных случаев АГ могут быть выделены микропрограммные АГ, где

$$\forall m \in M^T: e(m) = const$$

и булевые АГ, где

$$\forall m \in M^B: V(e(m)) = \{0,1\},$$
$$\forall m \in M^T: e(m) = const \in \{0,1\}.$$

Частный случай булевых АГ соответствует модели бинарных диаграмм, предложенной в [2].

Определим на модели АГ процесс моделирования (движение на графе по некоторому пути L, согласно заданным значениям переменных $z \in Z(L)$ и обратный этому процесс активизации некоторого пути L), нахождение значений $z \in Z(L)$, таких, для которых при моделировании реализовалось движение по заданному пути L. Все задачи диагностики дискретных устройств, связанные с синтезом и анализом тестов, приводимы к реализации этих двух процессов, т.е. к выбору и прохождению определенных путей на альтернативных графах.

2. Переменные модели

Весовые переменные вершины АГ классифицируются по структуре на простые переменные, где $n = 1$ (потенциальные и импульсные сигналы), слова, где $n > 1$ (параллельные и последовательные коды, последовательности импульсов) и массивы. По типу переменные модели могут быть классифицированы на функциональные переменные, предназначенные для описания чисто функциональных зависимостей рассматриваемого объекта, и специальные переменные (временные и пространственные переменные, переменные "рукопожатия" и др. вспомогательные переменные), предназначенные для задания дополнительных временных и пространственных соотношений при описании функций объекта.

Временные переменные введены для разделения рассматриваемого промежутка времени (командного цикла, машинного цикла, такта и т.д.) на более детальные части. Вершина m, взвешенная временной переменной $e(m) = t$ и имеющая более чем один выход $|V(e(m))| > 1$, свидетельствует о различном поведении переменной графа в различных частях рассматриваемого промежутка времени T, причем каждый выход вершины выделяет из этого промежутка одну определенную часть. Таким образом, задача вычисления значений переменной графа для всего промежутка T порождает некоторый процесс, при котором необходимо циклически продолжать движение по раз-

ным направлениям из рассматриваемой вершины. Временная переменная может и сама являться функцией от других переменных, описывая микропрограмму с ветвлениями и представляя собой состояние соответствующего микропрограммного автомата. Независимые временные переменные описывают стандартные циклы работы объекта.

Временных переменных может быть более одной. В этом случае образуется иерархия временных шкал. Если некоторый путь на графе проходит через несколько вершин с разными временными переменными, то первая пройденная вершина определяет время высшего уровня (например, такта), следующая - время низшего уровня (например, микротакта).

Функции, которые описывают поведение последовательных схем, опишем рекуррентными соотношениями

$$z_i = f_i(Z'),$$

где штрих при Z обозначает факт, что значения переменных $z \in Z$ берутся из "предыдущего момента времени". При иерархическом представлении времени "предыдущий момент" необходимо связывать с определенной шкалой времени (с уровнем). Условимся обозначать далее одним штрихом самый верхний уровень, двумя штрихами следующий нижний уровень и т.д. (см. рис. I).

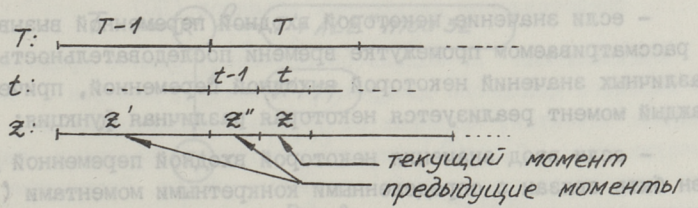


Рис. 1.

В качестве примера рассмотрим фрагмент АГ, описывающий выходное поведение шины данных микропроцессора INTEL 8080, приведенный на рис. 2. Здесь I обозначает байт команды, введенный в течение промежутка времени t = I (в первом машинном цикле), значение I = 34 соответствует команде STAX HL. В начале I-го цикла выдается константа #Fetch (слово состояния, характеризующее текущий машинный цикл), в

начале 2-го и 3-го циклов выдается константа #RD, в начале 4-го и 5-го циклов выдается константа #WR, а затем, соответственно, содержимые регистров H и L. Поведение шины соответствует некоторому обобщенному такту T, машинные циклы соответствуют следующему нижнему уровню времени, а промежутки τ и $\tau-1$ в 4-м и 5-м циклах соответствуют самому нижнему уровню времени. Штрих при H и L обозначает, что их значения соответствуют моменту, предыдущему этой команде.

Временные переменные целесообразно ввести в модель объекта в следующих случаях:

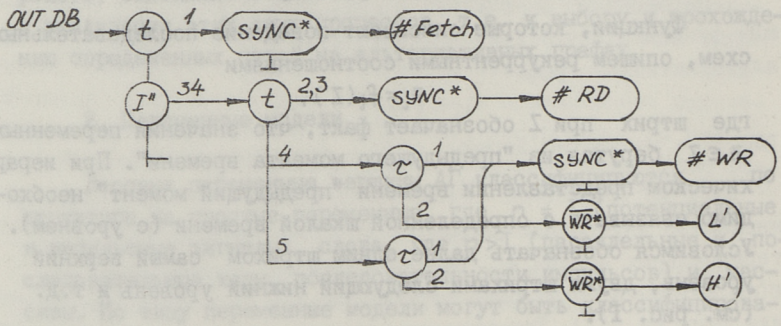


Рис. 2.

- если значение некоторой входной переменной вызывает в рассматриваемом промежутке времени последовательность различных значений некоторой выходной переменной, причем в каждый момент реализуется некоторая различная функция;

- если ввод значения некоторой входной переменной должен быть связан с определенными конкретными моментами (внутри рассматриваемого промежутка времени) или сигналами (например, сигналами "рукопожатия");

- если некоторая функция объекта в рассматриваемом промежутке времени не реализуется простым алгебраическим выражением, а требует представления в виде алгоритма, выполнение которого зависит от промежуточных результатов.

Введение вершин АГ с временными переменными позволяет:

- часть процессов (непрослеживаемые внутренние процессы в объекте) описывать грубее, часть же процессов (сеансы с внешним тестером) с требуемой степенью детальности;

- представлять произвольные микропрограммные модели в виде альтернативных графов.

Переменные "рукопожатия" вводятся в модель АГ с целью синхронизации процессов, происходящих в объекте, с тестовыми воздействиями, генерируемыми во внешнем тестере. Они позволяют задавать формальные условия для определения моментов, когда необходимо ввести информацию в объект или когда необходимо зафиксировать выходное поведение объекта. Так, например, на графе рис. 2 к переменным "рукопожатия" относятся SYNC* и $\overline{WR}$*, обозначенные звездочкой.

Если переменные "рукопожатия" представляют собой многоразрядные слова (параллельные или последовательные коды, последовательности импульсов), то из них составляют предикатные выражения. Так, например, при описании работы таймера в микропроцессоре INTEL 8048, условием прибавления в счетчик таймера очередной единицы является ALE* = 32, где ALE* - переменная "рукопожатия", представляющая импульсную пачку на выходе ALE. Модель АГ, описывающая работу таймера T, приведена на рис. 3. Здесь переменная R' определяет режим работы таймера (счет времени, счет внешних событий или режим покоя). Значение R' устанавливается командами STRT T, STRT CNT и STOP TCNT.

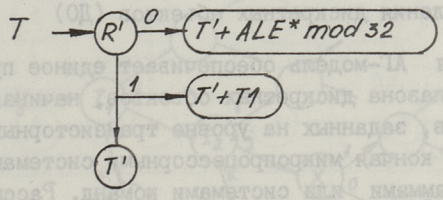


Рис. 3.

Пространственные переменные введены для избежания дублирования повторяющихся подфункций разных разрядов или полей формата вычисляемой на графе переменной. Вершины АГ, взвешенные пространственными переменными, определяют входы в разные подграфы, предназначенные для вычисления разных частей формата рассматриваемого слова (см. рис. 4). При стремлении к компактному представлению объекта, каждая вершина, взвешенная пространственной переменной дает эффект

$$E = \frac{kC + C_0}{C + C_0},$$

где C – цена подфункции f , выведенной из-под области влияния пространственной переменной;

C_0 – общая цена остальных подфункций f_i , $i = \overline{1,k}$.

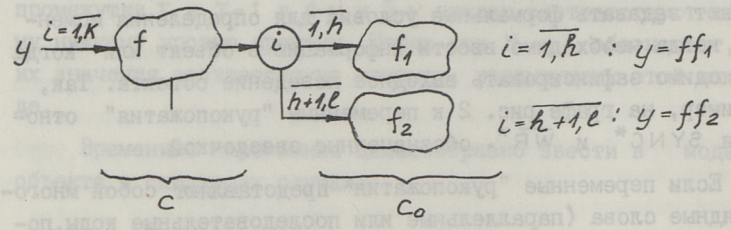


Рис. 4.

В частных случаях имеем:

$$E = k \quad \text{при} \quad C >> C_0,$$
$$E = 1 \quad \text{при} \quad C_0 >> C.$$

Критерием выбора вершин АГ с пространственными переменными служит очевидно $E = max$. В общем случае имеем дело со сложной комбинаторной задачей оптимизации декомпонирования функций объекта.

3. Сравнение АГ-модели с известными способами представления дискретных объектов (ДО)

Предложенная АГ-модель обеспечивает единое представление широкого диапазона дискретных объектов, начиная с нерегулярных объектов, заданных на уровне транзисторных или логических схем, и кончая микропроцессорными системами, заданными микропрограммами или системами команд. Рассмотрим ниже некоторые примеры.

Булевы уравнения непосредственных связей. Рассмотрим типичный операционный элемент, представленный в виде схемы и АГ на рис. 5 и реализующий некоторую функцию вида

$$y = F(I_1, I_2, X, S),$$

где I_1, I_2 – код команды заданный форматом из двух полей;
X – вектор условий;
S – данные.

Применение аппарата булевых функций требовало бы введения

фиктивных переменных I_{1j}, $j=\overline{1,n}$ и I_{2j}, $j=\overline{1,k}$, что приводит к записи

$$y = F(I_{11},\ldots,I_{1n},I_{21},\ldots,I_{2k},X,S).$$

При этом увеличение размерности модели сопровождается усложнением модели неисправностей, так как требуется выделение подмножеств переменных и отдельная спецификация неисправностей на уровне таких подмножеств. В случае АГ-модели промежуточные переменные $I_{11},\ldots,I_{2k}$ поглощаются и модель неисправностей не требует отдельного задания, а вытекает непосредственно из графа. Физическая суть различия этих двух подходов в отношении модели и задания неисправностей заключается в том, что на графе несколько дуг могут быть одновременно активизированы, а одна переменная в один и тот же момент времени нескольких значений иметь не может.

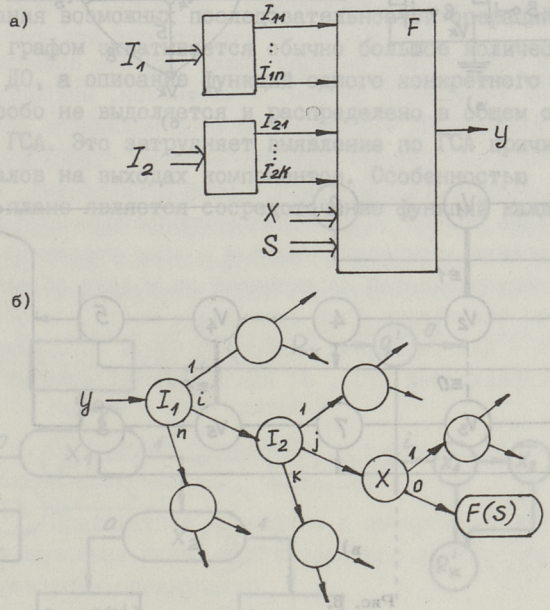


Рис. 5.

Двухполюсные ненаправленные графы (ДНГ). Известно, что традиционные методы синтеза тестов, базирующиеся на логических моделях ДО, не применимы для объектов, построенных на базе МОП-технологии [5]. Для выхода из положения в [5] предложена новая модель ДО на базе ДНГ, позволяющая ввести в модель и диагностировать топологические дефекты. Легко

заметить, что задачи синтеза и анализа тестов, решаемые на ДНГ, без труда переводимы и на язык АГ. Рассмотрим ДО на рис. 6 а.

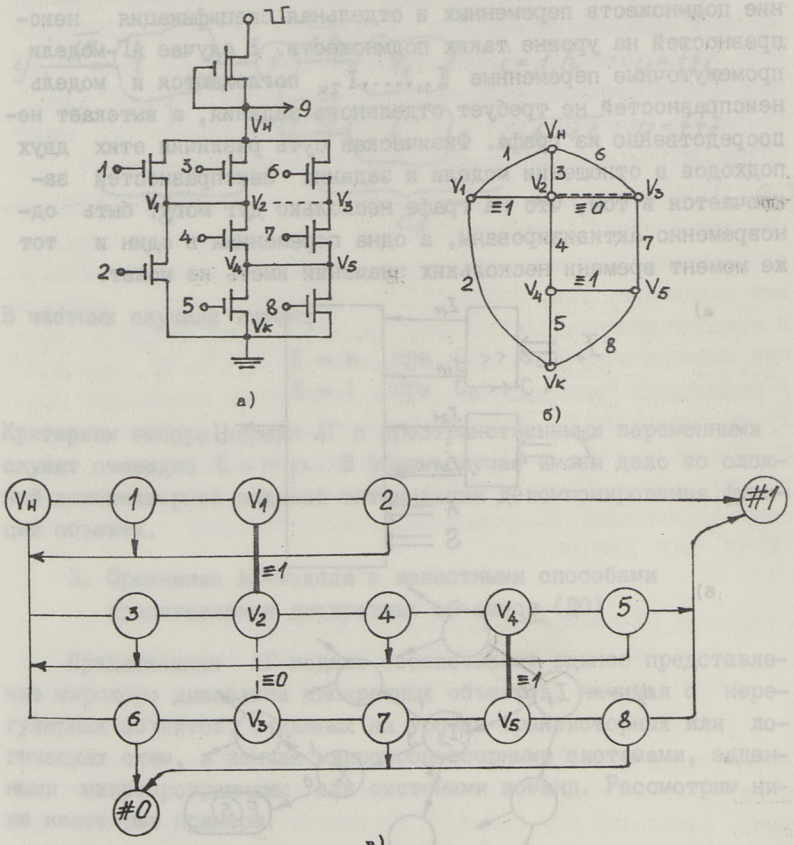


Рис. 6.

Соответствующие ему ДНГ и АГ приведены, соответственно, на рис. 6 б и 6 в. Ребра ДНГ представлены в АГ-модели вершинами. Для создания возможностей адекватного задания топологических дефектов объекта в АГ-модели сохранены вершины ДНГ, выходы которых считаются постоянно активизированными. С этой же целью все дуги графа должны оставаться двунаправлен-

ными, что эквивалентно появлению при вершинах двух всегда одновременно активизируемых I-выходов (выходов, активизированных при e(m)= I). Введение топологических неисправностей типа обрыва и короткого замыкания в АГ-модель соответствует снятию и введению условия постоянной активизации при дугах АГ.

Таким образом, путем введения определенных ограничений в АГ-модель оказывается возможным с одной методологической позиции решать на ней как традиционные задачи диагностирования ДО, заданных логическим уравнением [4], так и новые задачи диагностирования МОП-схем, заданных своей топологией, рассматриваемые в [5].

Граф-схемы алгоритмов (ГСА). ГСА предназначены для описания возможных последовательностей операций в ДО. При этом графом охватывается обычно большое количество аппаратуры ДО, а описание функций одного конкретного компонента ДО особо не выделяется и распределено в общем случае по всей ГСА. Это затрудняет выявление по ГСА причин ложных сигналов на выходах компонентов. Особенностью АГ-модели в этом плане является сосредоточение функций каждого компо-

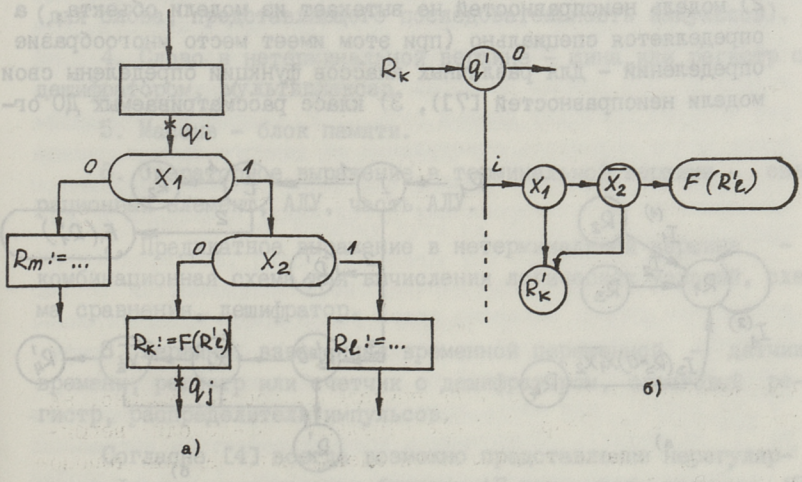


Рис. 7.

нента ДО в отдельном графе. Из этого следует возможность непосредственного определения причин ложного поведения компонента, в любой момент выполнения микропрограммы. Так, например, на рис. 7 иллюстрирован фрагмент ГСА и соответствующий компоненту R_K фрагмент АГ-модели. При ложном сигнале на R_K следует анализировать соответствующий путь на АГ, согласно значениям весовых переменных. При этом возможно сохранение соответствия между АГ-моделью и структурой: q' - регистр адреса ПЗУ с дешифратором, x_1, x_2 - комбинационная часть автомата, $F(R_i)$ - функция операционного автомата. Заметим, что структурная таблица микропрограммного автомата [6] может быть рассмотрена как частный случай АГ, причем последние отличаются более сжатым представлением информации. Одна строка в структурной таблице соответствует одному пути на АГ.

Модели регистровых передач нашли широкое применение при описании дискретных систем, в частности, микропроцессорных БИС и систем [7]. Недостатками этой модели по сравнению с АГ являются: 1) (неоднородность модели) требуются три компонента различного характера - граф регистровых передач (ГРП) описание команд (информация микропрограммного характера) и описание логических условий в виде уравнений, 2) модель неисправностей не вытекает из модели объекта, а определяется специально (при этом имеет место многообразие определений - для различных классов функций определены свои модели неисправностей [7]), 3) класс рассматриваемых ДО ог-

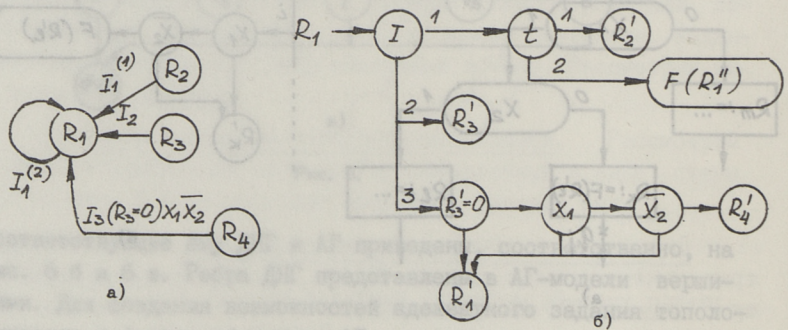


Рис. 8.

раничен только микропроцессорами. В АГ-модели ГРП соответствует система АГ непосредственных связей [4], где каждой вершине ГРП соответствует свой АГ, задающий в однородном виде всю информацию микропрограммного или логического характера, для которого в ГРП-модели требуются дополнительные неграфовые средства. Пример сравнения ГРП и АГ приведен на рис. 8.

4. Представление структуры объекта и неисправностей альтернативными графами

Модель АГ может быть использована как для представления функций дискретных объектов, так и для представления их структуры. Рассмотрим некоторые возможности структурной интерпретации вершин АГ, взвешенных теми или иными типами переменных или выражениями на основе их.

I. Константа в терминальной вершине - ячейка ПЗУ.

2. Вершина, взвешенная простой переменной - сигнал, точка или путь в схеме, точка разветвления, триггер.

3. Слово в терминальной вершине - шина, регистр, ячейка памяти (для параллельного кода), путь в схеме, сдвиговый регистр (для последовательного кода), путь в схеме, счетчик (для слова, представляющего последовательность импульсов).

4. Слово в нетерминальной вершине - шина или регистр с дешифратором, мультиплексер.

5. Массив - блок памяти.

6. Операторное выражение в терминальной вершине - операционный элемент, АЛУ, часть АЛУ.

7. Предикатное выражение в нетерминальной вершине - комбинационная схема для вычисления логических условий, схема сравнения, дешифратор.

8. Вершина, взвешенная временной переменной - датчик времени, регистр или счетчик с дешифратором, сдвиговый регистр, распределитель импульсов.

Согласно [4] всегда возможно представление нерегулярных цифровых схем моделью булевых АГ так, чтобы вершины АГ представляли определенные пути в исходной схеме. Свойства

регулярности цифровых схем (шин, регистров, памяти, дешифраторов, мультиплексеров и т.д.) могут быть отражены в модели АГ в структуре переменных и в структуре выходов соответствующих вершин. При отсутствии информации о структуре нерегулярных схем или при желании уменьшить размерность модели объекта, определенные функционально-законченные блоки или подсхемы могут быть представлены единственной вершиной взвешенной соответствующими выражениями. Следовательно, при представлении дискретных устройств альтернативными графами всегда имеется возможность регулировать сложность модели и точность задания в ней структуры в двух направлениях:

- увеличить число вершин с одновременным упрощением весовых выражений при вершинах;

- уменьшить число вершин с одновременным введением функций, соответствующих удаляемым подграфам, в весовые выражения других вершин.

Так, например, подграф, представляющий структуру сумматора с точностью до путей в ее логической схеме и вершины которого взвешены булевыми переменными, можно заменить единственной вершиной, взвешенной выражением суммы.

Каждый путь в АГ отражает определенный режим работы описываемого графом компонента объекта. Неисправности, могущие повлиять на рассматриваемый режим, описываются на модели АГ при помощи неисправностей на выходах вершин, через которые проходит данный путь. Так, например, неисправность может порождать ответвление от данного пути к некоторой другой конечной вершине или генерировать несколько одновременно активизированных путей между начальной и конечными вершинами.

Традиционная модель константных неисправностей на вентильном уровне заменена здесь более общей моделью константных неисправностей на ветвях графов. Вершине графа может иметь обрыв на какой-то выходной ветви (константа "0" на ветви) или иметь постоянно открытый выход (константа "1" на ветви). Допустимы и кратные дефектные ветви.

Рассмотрим вершину m с весовым выражением $e(m)$ и с областью определения $V(e(m))$. Сопоставим с каждой ветвью

вершины m булевую переменную v_i, $i \in V(e(m))$, значение которой будем интерпретировать следующим образом

$$v_i = \begin{cases} 0, & \text{если } e(m) \neq i, \\ 1, & \text{если } e(m) = i. \end{cases}$$

Тогда с ветвями v_i вершин АГ могут быть привязаны следующие неисправности:

I) $v_i \equiv 0$ (соответствующая ветвь v_i в АГ отсутствует);

2) $v_i \equiv 1$ (соответствующая ветвь v_i постоянно активизирована);

3) $v_i \to v_j$ (вместо ветви v_i активизируется ветвь v_j);

4) $v_i \to v_i, v_j$ (дополнительно к ветви v_i активизируется еще ветвь v_j).

Для интерпретации предложенной модели рассмотрим некоторые примеры. Для булевых переменных $e(m) \in \{0,1\}$ моделью покрываются константные неисправности $e(m) \equiv 0$ и $e(m) \equiv 1$. Для многоразрядных слов, взвешивающих нетерминальные вершины и представляющих, например, коды команд, коды состояний микропрограммных автоматов, моделью покрываются функциональные дефекты дешифраторов и управляемых ими мультриплексеров, в частности, ею покрывается модель неисправностей, введенная в работах [7] для микропроцессоров. Для весов, представляющих собой выражения $f(Z)$ или $p(Z)$, моделью покрываются функциональные дефекты соответствующих блоков. В зависимости от точности отражения на модели внутренней структуры объекта, предложенная модель неисправностей позволяет покрывать широкий класс структурных и функциональных неисправностей дискретных устройств, имеющих практическое значение.

В общем случае каждому пути на модели АГ соответствует список дефектов, рассматриваемых как потеря сигнала (константы "0" на ветвях пути) или появление нового сигнала (константы "I" на ветвях из вершин пути). Если мы знаем на основе информации о внутренней структуре объекта, что некоторые функционально несвязанные сигналы в результате дефекта могут влиять друг на друга (например, случаи коротких замыканий или перепутывания связей, случаи чувствительности наборов и т.д.), мы можем формально учитывать эти дефекты в модели, вставляя в графы дополнительные вершины, связывающие данные сигналы. Так, например, пусть для схемы на рис.9

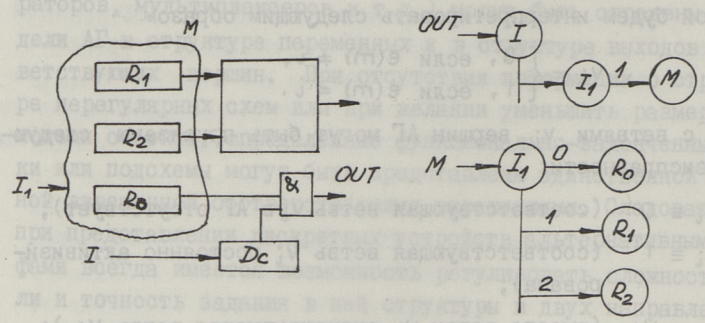


Рис. 9.

известно, что содержимое R_0 из-за структурной "близости" может быть искажено содержимыми R_1 и R_2. Для учета этих неисправностей в модель введены фиктивная магистраль M и фиктивная переменная управления I_1. Неисправности вершины с переменной I_1 в графе M отображают теперь указанные физические дефекты влияния R_1 и R_2 на R_0.

Пример представления дефектов цифровой системы на модели АГ рассматривается на рис. I0.

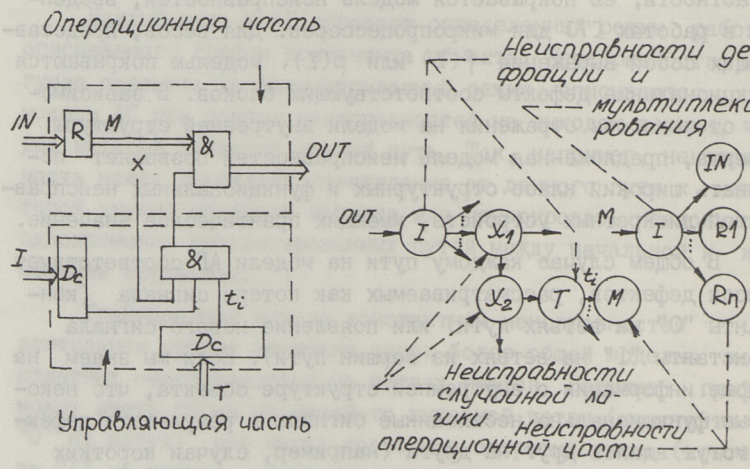


Рис. 10.

5. Генерирование тестов

Генерирование тестов для дискретных объектов, в частности, микропроцессорных БИС и систем включает следующие этапы:

- синтез последовательности команд (тест-программы), предназначенных для загрузки и транспортировки нужных операндов, для проведения проверяемой операции и для транспортировки результатов теста к выходу объекта;

- синтез таких операндов, чтобы удовлетворялись условия для обнаружения заданных дефектов.

На модели АГ задача синтеза тестов формулируется как задача синтеза тестов для проверки вершин АГ. Вершины, подлежащие проверке, подбираются в процессе последовательной активизации путей, проводимом так, чтобы все вершины в графе оказались пройденными. Пусть L - текущий активизированный путь в графе G_K из начальной вершины к некоторой терминальной вершине m^T, проходящий некоторое подмножество вершин $L \subseteq M_K$. Если существует хотя бы одна непроверенная вершина $m \in L$, для нее строится тест. При этом может быть использовано следующее утверждение.

Утверждение I. Вершина m считается проверенной, если построены тесты, проверяющие все неисправности $v_i \equiv 0$, где $i \in V(e(m))$, и неисправности $v_i \equiv 1$ при условиях $e(m)=j$, где $i, j \in V(e(m))$.

Рассмотрим граф G_K, представляющий некоторую функцию $z_K = f_K(Z)$. Обозначим через $L_i(m)$ путь между начальной вершиной и некоторой терминальной вершиной $m_i^T \in L_i(m)$, проходящий вершину m по ветви v_i. Легко убедиться в правильности следующего утверждения.

Утверждение 2. Тест, который активизирует путь $L_i(m)$, обнаруживает совокупность неисправностей

$$\{ v_i \equiv 0,\ v_i \equiv 1 \mid j = \overline{1,h};\ j \neq i \},$$

где $h = |V(e(m))|$, если им активизированы также пути $L_j(m)$, $j = \overline{1,h}, j \neq i$, так, чтобы удовлетворялось условие

$$\forall j,\ j = \overline{1,h}, j \neq i : e(m_i^T) \neq e(m_j^T).$$

На основе этого утверждения базируется формальный алгоритм генерирования тестов, использующий модель АГ.

Легко заметить, что удовлетворение усиленного условия

$$\forall i,j \in V(e(m)),\ i \neq j : e(m_i^T) \neq e(m_j^T)$$

позволяет организовать тест, проверяющий одновременно все выходы вершины m. Для этого необходимо повторить тест h раз, изменяя циклически значение $e(m)$ и реализуя таким образом все ветвления из вершины m.

Итак, учитывая вышесказанное, синтез теста для некоторого выхода v_i вершины m состоит из следующих этапов:

- активизация некоторого пути из начальной вершины графа до проверяемой вершины m;

- активизация несовпадающих путей для каждого выхода v_i вершины m до терминальных вершин m_i^T;

- решение системы неравенств

$$e(m_i^T) \neq e(m_j^T),\ j = \overline{1,h},\ j \neq i,$$

где h - число выходов вершины.

В случае микропрограммных и булевых АГ, в которых терминальные вершины взвешены константами, второй и третий этапы совмещаются. В случае операционных устройств типа ЭВМ или микропроцессор на первом и втором этапах синтезируется тест-программа, а на третьем этапе - операнды для этой программы.

При проверке терминальных вершин требуется решение всего лишь первого этапа. Решение второго и третьего этапов сводится к циклическому повторению теста, найденного на первом этапе, для различных значений проверяемого весового выражения $e(m^T)$. Эти значения могут быть найдены методом полного перебора или путем генерирования случайных чисел.

6. О синтезе тестовых групп

Достоверность получаемых тестов существенно зависит от того, синтезированы ли они для случая единственной неисправности или для случая произвольной кратной неисправности. В работе [8] был предложен метод синтеза тестовых групп для комбинационных схем, базирующийся на модели булевых АГ,

и доказана полнота получаемых тестов относительно кратных константных неисправностей. В данной работе результаты, полученные в [8], обобщаются на общий случай АГ.

Целью составления тестовых групп является не проверка отсутствия неисправностей на ветвях вершины АГ, а проверка исправности путей АГ в условиях кратных неисправностей. Для путей активизации определенного пути L_i составляется некоторый тестовый набор (например, команда микропроцессора, состояние программируемого интерфейса и т.д.), а также некоторая программа, включающая устанавливающую часть, тестовый набор и транспортирующую часть. Программа должна выполняться циклически так, чтобы при каждом ее выполнении изменилось значение веса при некоторой вершине $m \in L_i$ и чтобы в конечном итоге активизировались циклически все выходы вершин на проверяемом пути. Каждая такая модификация программы должна выполняться в общем случае многократно с операндами, выбранными так, чтобы ответвления от проверяемого пути дали бы результаты, отличные от базового варианта (от выполнения проверяемого пути L_i).

Доказательство полноты получаемых тестов относительно кратных неисправностей из установленного класса на рассматриваемом графе можно провести аналогично доказательству для частного случая булевых АГ [8].

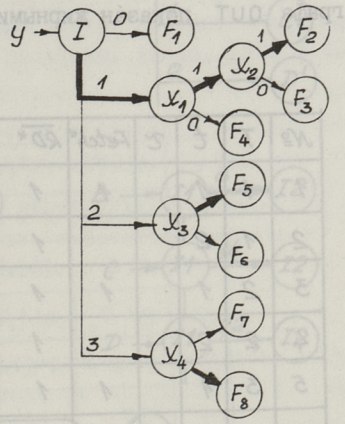


Рис. 11.

В качестве примера рассмотрим проверку пути $I = 1$, $x_1 = I$, $x_2 = I$, F_2 на АГ, представленном на рис. II. Здесь вершины I и x_i описывают некоторый управляющий автомат (I – регистр команды с дешифратором, x_i – триггера условий), вершины F_j – функциональные компоненты некоторого операционного автомата. Для проверки вершины I активизированы дополнительные пути $x_3 = I$, $x_4 = 0$ и удовлетворены условия

$F_2 \neq F_1$, $F_2 \neq F_5$ ∪ $F_2 \neq F_8$.

Для проверки вершин x_1 и x_2 необходимо решение неравенств, соответственно, $F_2 \neq F_4$ и $F_2 \neq F_3$. В таблице I представлен обобщенный набор $T_D = DDD10D(I, x_1, x_2, x_3, x_4, y)$, необходимый для проверки рассматриваемого пути (через D обозначены изменяемые циклически переменные), а также конкретные реализуемые наборы $T_1 - T_6$. Обобщенный набор T_D может быть рассмотрен как сжатое представление множества наборов $T_1 - T_6$.

Таблица I

T	I	x_1	x_2	x_3	x_4	y
T_D	D	D	D	1	0	D
T_1	0	1	1	1		F_1
T_2	1					F_2
T_3	2					F_5
T_4	3					F_8
T_5	1	0				F_4
T_6		1	0			F_3

Пример построения теста для части функций аккумулятора микропроцессора INTEL 8080 рассматривается на рис. I2 и I3, результаты приведены в табл. 2 и 3. На рис. I2 представлен фрагмент модели АГ для микропроцессора. На графе OUT показан жирными линиями путь, соответствующий

Таблица 2

№	T	t	τ	Fetch*	$\overline{RD}$*	$\overline{WR}$*	IN DB	OUT DB	Действие
1	1	1		1	1		6		MVI B, data
2	1	2			1		B'		Ввод B'
3	2	1		1	1		7.6		MVI A, data
4	2	2			1		A'		Ввод A'
5	3	1		1	1		VAR(i)		Вариация
6	3	2			1		IN2		Ввод IN2
7	3	3			1		IN3		Ввод IN3
8	4	1		1	1		2		STAX
9	4	2	2			1		ET(i)	Ввод эталона

выполнению команды STAX, необходимой для транспортировки содержимого аккумулятора A (результата теста) на выходную шину данных. Граф A задает поведение аккумулятора при выполнении некоторого подмножества команд (см. табл. 3). Граф R определяет участвующий при этих командах регистр. Графы B, C и D описывают функции, необходимые для реализации установочных операций. На графах I1, I2 и I3 определяются условия для ввода информации в микропроцессор. По-

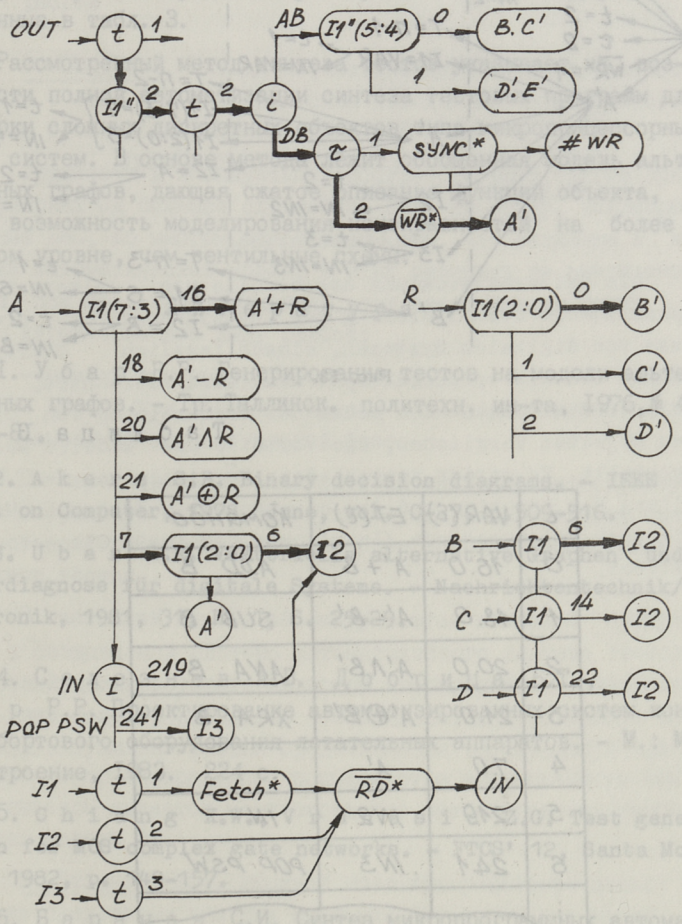


Рис. 12.

следовательность прохождения путей на графах иллюстрируется на рис. I3. Невисячие вершины этого дерева представляют графы, а их последователи соответствуют вершинам, пройденным на графах. На основе дерева сформирована программа, представленная в таблице 2, где каждой строке соответствует определенный момент времени. Промежутки T = I,2 соответству-

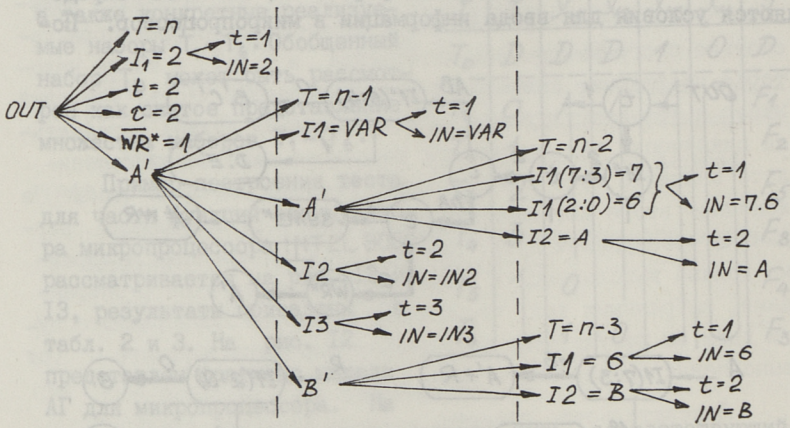


Рис. 13.

Таблица 3

i	VAR(i)	ET(i)	Команда
0	16.0	A'+B'	ADD B
1	18.0	A'-B'	SUB B
2	20.0	A'∧B'	ANA B
3	21.0	A'⊕B'	XRA B
4	7.0	A'	
5	219	IN2	IN
6	241	IN3	POP PSW
⋮			

ют установке микропроцессора в необходимое исходное состояние, промежуток T = 3 соответствует проведению теста и промежуток T = 4 соответствует транспортировке результата теста на выход. Для проверки всех выходов вершины I1(7:3) в графе A необходимо выбирать операнды A и R так, чтобы выполнялось условие

$$A' + B' \neq A' - B' \neq A' \wedge B' \neq A' \oplus B' \neq A' \neq IN2 \neq IN3.$$

Программа, приведенная в табл. 2, выполняется циклически 7 раз, реализуя поочередно в промежутке T = 3 команды, перечисленные в табл. 3.

Рассмотренный метод синтеза тестов указывает на возможности полной автоматизации синтеза тестовых программ для проверки сложных дискретных объектов типа микропроцессорных БИС и систем. В основе метода лежит обобщенная модель альтернативных графов, дающая сжатое описание функций объекта, а также возможность моделирования неисправностей на более высоком уровне, чем вентильные схемы.

Литература

I. Убар Р.Р. Генерирование тестов на модели альтернативных графов. - Тр. Таллинск. политехн. ин-та, 1976, № 409, с. 75-81.

2. Akers S.B. Binary decision diagrams. - IEEE Trans. on Computer, 1978, June, vol. C-27, p.509-516.

3. Ubar R.R. Vektorielle alternative Graphen und Fehlerdiagnose für digitale Systeme. - Nachrichtentechnik/ Elektronik, 1981, 31, H. 1, S. 25-29.

4. Селезнев А.В., Добрица Б.Т., Убар Р.Р. Проектирование автоматизированных систем контроля бортового оборудования летательных аппаратов. - М.: Машиностроение, 1983. 224 с.

5. Chiang K.W., Vranesic Z.G. Test generation for MOS complex gate networks. - FTCS' 12, Santa Monika, 1982, p. 149-157.

6. Баранов С.И. Синтез микропрограммных автоматов. - Л.: Энергия, Ленинградское отделение, 1979. 232 с.

7. Thatte S.M., Abraham I.A. Test generation for microprocessors. - IEEE Trans. on Computer, 1980, June, vol. C-29, p. 429-441.

8. Убар Р.Р. Построение полных тестов для комбинационных схем. - Изв. АН ЭССР, том 31, Физика/Математика, 1982. № 4, с. 418-427.

R. Ubar

Universaler Zugang zur Automatisierung von Testentwicklung für breite Klasse von digitalen Objekten

Zusammenfassung

Die Weiterentwicklung des Modells von alternativen Graphen zwecks Formalisierung der Testsynthese für breite Klasse von digitalen Systemen, einschliesslich Mikroprozessoren und Mikroprozessorsystemen, wird betrachtet. Das Modell wird mit bekannten Arten der Beschreibung von digitalen Objekten verglichen, die Vorteile des Modells werden dargestellt. Es werden verschiedene Möglichkeiten der Darstellung der Struktur und der strukturellen Fehler im Modell sowie auch entsprechende Algorithmen der Testsynthese beschrieben.

№ 626

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.3

В.Р. Аланго, Т.К. Коньт

ПРЕПРОЦЕССОР К СИСТЕМЕ АВТОМАТИЧЕСКОГО СИНТЕЗА ТЕСТОВ

I. Введение. Методы автоматического синтеза диагностических тестов для цифровых схем исходят из математической модели объекта диагностирования (ОД).

Трудности в эксплуатации соответствующих систем при нерегулярных ОД большой сложности, как, например, микропроцессорные БИС, во многом обусловлены тем, что создание и кодировка математической модели в этих системах не автоматизированы. Как показывает практика, при эксплуатации таких систем характерны два вида ошибок:

- ошибки при кодировке и вводе;
- ошибки при создании модели и в предшествующей этому так называемой концептуальной фазе.

По мере повышения сложности ОД значение ошибок второго типа возрастает и, что еще хуже, среди них с большой вероятностью встречаются такие, которые не может обнаружить синтезатор тестов.

В настоящей статье рассматривается языковой препроцессор к создаваемой системе автоматического синтеза тестов [I]; основной целью которого является автоматизация вышеупомянутых двух фаз. Препроцессор обеспечивает интерактивную работу на концептуальном уровне представления ОД, используя при этом проблемно-ориентированное подмножество естественного языка. Его выходом является соответственно закодированная математическая модель ОД.

2. Входной язык. Входной язык препроцессора должен обладать средствами как для описания ОД, так и для ввода запросов и директив для управляющего монитора.

При описании сложных цифровых объектов, в частности, микропроцессорных БИС и систем, является целесообразным описывать объект исходя из нескольких разных аспектов (структурного, функционального) и на разных уровнях (например, микропрограмма, режим работы и т.д.). При этом накапливается избыточная информация, которая может оказаться полезной при поиске логических ошибок в описании ОД, при включении данного объекта в какую-нибудь крупную структуру или при конечном оформлении результатов работы генераторов тестов.

Однако, для такого подхода характерно резкое усложнение как языка описания ОД, так и языка опросов (составление и редактирование концептуальной модели ОД является интерактивным процессом, организуемым диалоговыми средствами препроцессора). Отсюда вытекает требование к расширяемости входного языка.

В связи с вышеуказанными обстоятельствами мы отказываемся от использования "четкого" (в синтаксическом смысле) входного языка и вместе с тем используем сильно ограниченное подмножество естественного языка. Многие вопросы такого подхода основательно проработаны [2].

Входной язык является в большей степени проблемно-ориентированным. Это выражается в том, что его лексика весьма ограничена:

- существительными являются преимущественно специфические термины, как бит, регистр, состояние, очередь и т.д.;

- глаголы ограничиваются формами "быть", "иметь", плюс команды к монитору, например, "печатать", "загружать";

- прилагательные отсутствуют.

Поэтому и соответствующие группы слов и предложения в целом имеют также простой вид. Примеры диалога при создании концептуальной модели для микропроцессора К 580ИК80 представлены на рис. I. Символы, выпечатанные ЭВМ, в примере подчеркнуты.

3. Концептуальная модель ОД. Концептуальная модель (КМ) соединяет в одно логическое целое все вводимые пользователем описания ОД, составленные исходя из разных аспектов. Надо отметить, что сущность КМ намного богаче, чем просто формальное представление ОД - в ней описываются все понятия и

```
  ...
>USE RP FOR REGISTER PAIR :
WHAT IS THE REGISTER PAIR ?   CONCATENATION (RP_H RP_L)
>TABLE OF RP (
(NAME RP_H RP_L CODE)
( WZ    W    Z   NIL)
( BC    B    C   ØØ )
   ...
( SP  SPH  SPL   LL ))
>INSTRUCTION TABLE
((NAME       FORMAT   CODE   OPERATIONS)
((MOV DST,SRC)   1  Ø1DDDSSS (DST = SRC))
( XCHG           1  111Ø1Ø11 (HL == DE ))
    ... )
UNDEFFINED FIELD D ?
>DDD IS CODE OF DST; SSS IS CODE OF SRC .
>LIST REGISTER NAMES :
(A B C D E H L)
>  ...
```

Рис. 1. Примеры диалога.

термины, введенные пользователем. В законченном КМ содержится вся информация, необходимая для создания математической модели ОД.

КМ реализуется в виде так называемой семантической сети, что представляет собой ориентированный граф, с помеченными вершинами и дугами, причем вершинам соответствуют примитивы (элементарные объекты, понятия), а дугам - отношения между ними [3].

В конкретной реализации сети характер отношения описывается при помощи специального признака при каждой дуге - дескриптора связи; а особые свойства каждого примитива - при помощи атрибутов. Внутреннее представление семантической сети в данной реализации несколько модифицировано - допускается ветвление дуг (используя псевдовершины). Значение исходных примитивов (словарных понятий), атрибутов и дескрипторов зафиксировано программно.

В момент запуска программа КМ пуста и все примитивы содержатся во вспомогательном - словарном списке. Но в

этот момент у них уже имеются определенные атрибуты и дескрипторы так называемых обязательных связей, т.е. связей, создание которых обязательно при включении данного примитива в состав КМ. Этот механизм позволяет автоматически следить за цельностью КМ. С помощью атрибутов проверяется "соответствие типов" связанных примитивов, или другими словами, отсутствие логических противоречий в КМ.

По мере включения примитивов в структуру КМ исчезает разница между словарем и КМ, так как примитивы при этом не дублируются.

Примеры представления примитива в словаре и его связей в составе КМ представлены на рисунках 2 и 3.

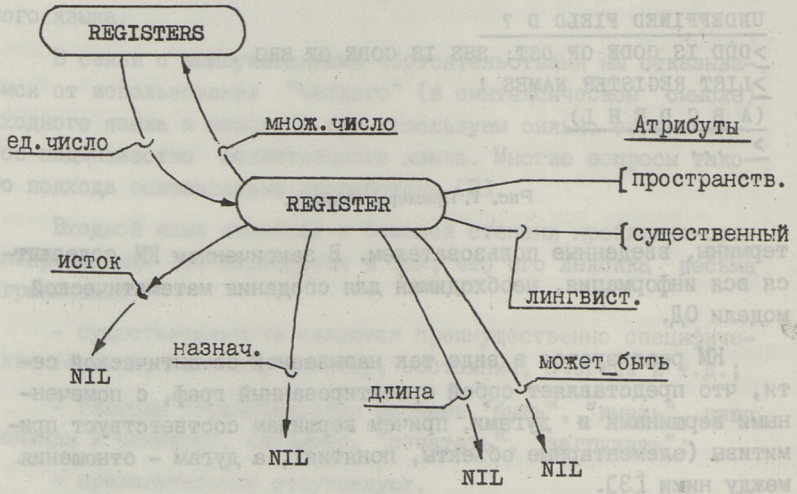


Рис. 2. Представление примитива в словаре.

Как видно из рис. 3, с существительным "REGISTER" связываются все общие "качества" отдельных его реализаций (имена существительные A,B,...,L). Индивидуальные качества каждой реализации связываются только с ней. Если какая-нибудь из обязательных связей реализуется в пределах группы реализации по-разному, то необходимо описать ее отдельно для каждого члена, а соответствующую связь от наименования группы направить к специальному символу (на рис. 3 - символ "*").

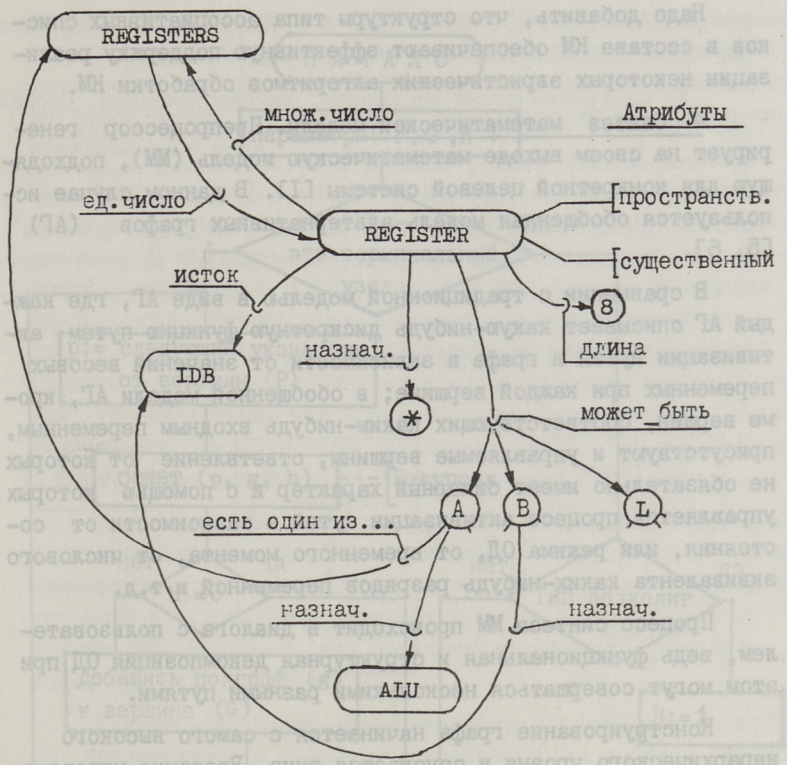


Рис. 3. Пример фрагмента КМ.

В данной реализации семантической сети допускается многозначность слов в случае, если разные его значения относятся к разным (в лингвистическом смысле) классам. Например, слово "A" - неопределенный артикль в английском языке, а в наших примерах оно служит также для обозначения одного из регистров ОД, т.е. является именем собственным.

При отражении отношений в пределах многих изоморфных подструктур КМ используется представление в виде ассоциативного списка. В конкретном случае это список, первым элементом которого является подсписок маски запроса, а остальными - подсписки конкретных объектов. Ассоциативные списки в составе КМ по существу соответствуют внешнему представлению в виде таблиц (см. рис. I).

Надо добавить, что структуры типа ассоциативных списков в составе КМ обеспечивают эффективную поддержку реализации некоторых эвристических алгоритмов обработки КМ.

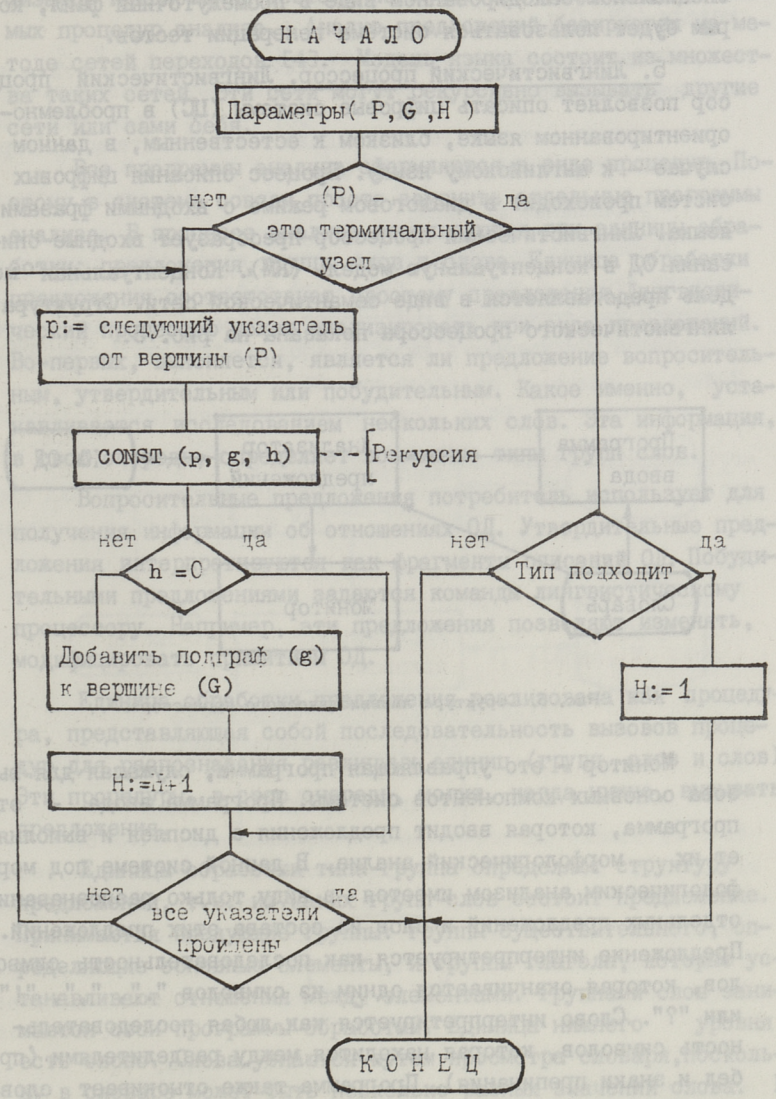
4. Синтез математической модели. Препроцессор генерирует на своем выходе математическую модель (ММ), подходящую для конкретной целевой системы [1]. В данном случае используется обобщенная модель альтернативных графов (АГ) [5, 6].

В сравнении с традиционной моделью в виде АГ, где каждый АГ описывает какую-нибудь дискретную функцию путем активизации путей в графе в зависимости от значений весовых переменных при каждой вершине; в обобщенной модели АГ, кроме вершин, соответствующих каким-нибудь входным переменным, присутствуют и управляемые вершины, ответвление от которых не обязательно имеет бинарный характер и с помощью которых управляется процесс активизации путей в зависимости от состояния, или режима ОД, от временного момента, от числового эквивалента каких-нибудь разрядов переменной и т.д.

Процесс синтеза ММ происходит в диалоге с пользователем, ведь функциональная и структурная декомпозиция ОД при этом могут совершаться несколькими разными путями.

Конструирование графа начинается с самого высокого иерархического уровня и спускается вниз. Введение управляемых вершин числового и временного типа происходит автоматически, потому что списковые структуры в КМ (например, ветвление к конкретным регистрам из вершины "REGISTER" на рис. 3, ассоциативный список (таблица) команд на рис. 1) трактуются как упорядоченные множества объектов. Управляемые вершины пространственного типа (ветвление по значению полей) конструируются при встрече с обозначением альтернативных полей (например, "ØI $_{dddsss}$") при прохождении списка в КМ. При многочисленных разветвлениях из управляемых вершин каждый следующий подграф снабжается индивидуальным именем и конструируется таким же образом.

Процедура просматривания структур КМ и построения подграфов ММ является рекурсивной. Ее значительно упрощенная блок-схема изображена на рис. 4. Периодически (по мере исчерпания памяти) проводится сравнение подграфов и сокращение их числа при совпадении. Созданные АГ записываются в



P - указатель на КМ
G - указатель на ММ
H - оценка удачи

Рис. 4. Рекурсивная процедура "CONST" построения ММ.

специальном закодированном виде в промежуточный файл, которым будет пользоваться система генерации тестов.

5. Лингвистический процессор. Лингвистический процессор позволяет описать цифровые системы (ЦС) в проблемно-ориентированном языке, близком к естественным, в данном случае - к английскому языку. Процесс описания цифровых систем происходит в диалоговом режиме с входными фразами языка. Лингвистический процессор преобразует входные описания ОД в концептуальную модель (КМ). Концептуальная модель представляется в виде семантической сети. Структура лингвистического процессора показана на рис. 5.

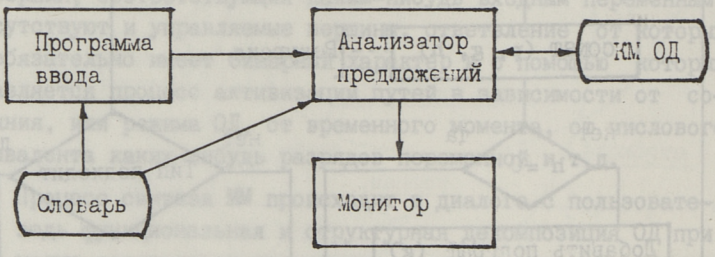


Рис. 5. Структура лингвистического процессора.

Монитор - это управляющая программа, служащая для вызова основных компонентов системы. Программа ввода - это программа, которая вводит предложения с дисплея и выполняет их морфологический анализ. В данной системе под морфологическим анализом имеется в виду только распознавание отдельных предложений и слов из состава этих предложений. Предложение интерпретируется как последовательность символов, которая оканчивается одним из символов ".", ",", "!" или "?". Слово интерпретируется как любая последовательность символов, которая находится между разделителями (пробел и знаки препинания). Программа также отыскивает слова в словаре.

Анализатор предложений является основным компонентом лингвистического процессора. Этот компонент производит синтаксический, семантический и концептуальный анализ предложений и генерирует КМ ОД. КМ ОД сохраняются в архиве. Ана-

лизатор предложений состоит из множества рекурсивно вызываемых процедур анализа. Анализ предложений базируется на методе сетей переходов [4]. Модель языка состоит из множества таких сетей. Эти сети могут рекурсивно вызывать другие сети или сами себя.

Все программы анализа оформляются в виде процедур. Поэтому в системе совсем просто заменить отдельные программы анализа. В процессе анализа используются три единицы обработки: предложения, группы слов и слова. Единица обработки предложения соответствует простому предложению. Лингвистический процессор может анализировать три вида предложений. Во-первых, выясняется, является ли предложение вопросительным, утвердительным или побудительным. Какое именно, устанавливается исследованием нескольких слов. Эта информация, в свою очередь, определяет возможные типы групп слов.

Вопросительные предложения потребитель использует для получения информации об отношениях ОД. Утвердительные предложения интерпретируются как фрагменты описаний ОД. Побудительными предложениями задаются команды лингвистическому процессору. Например, эти предложения позволяют изменять, модифицировать примитивы ОД.

Единица обработки предложения реализована как процедура, представляющая собой последовательность вызовов процедур для распознавания различных единиц (групп слов и слов). Эти процедуры, в свою очередь, могут, когда нужно, вызывать предложения.

Единицы обработки типа группа определяют структуру предложения, т.е. из каких групп слов состоит предложение. Принимаются следующие группы: группы существительного, определяющие основные элементы, и группы глагола, которые устанавливают отношения между элементами. Группами слов занимаются свои программы обработки. Единица нижнего уровня есть слово. Слова узнаются путем просмотра словаря, поскольку в словаре может быть несколько разных значений слова. В словаре находятся примитивы ОД и команды лингвистического процессора. С каждым словом в словаре связаны признаки синтаксического и семантического анализа.

6. Реализация. Рассмотренный процессор представляет собой пакет программ, работающих под управлением общего монитора и выполняющих следующие основные функции:

- создание, просмотр и редактирование концептуальной модели (КМ) ОД;

- автоматический контроль логической целостности и непротиворечивости КМ;

- синтез математической модели в виде системы альтернативных графов, исходя из КМ ОД;

- управление архивом.

Полученная математическая модель передается системе генерации тестов с помощью буферного файла. Архив служит для хранения как справочной информации о разных ОД, так и переработанной информации в виде законченных КМ или их фрагментов.

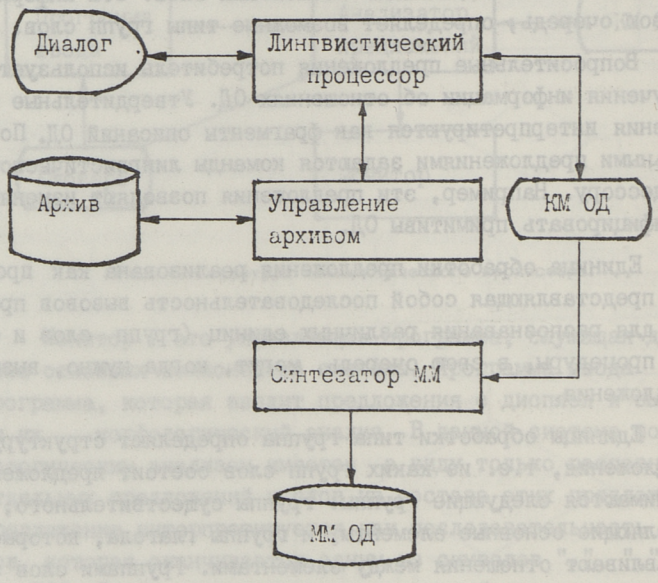


Рис. 6. Инфологическая схема препроцессора.

Инфологическая схема препроцессора показана на рис. 6.

В первичном варианте пакет реализуется на ЭВМ СМ-4 под ОС-РВ на языке LISP.

Литература

I. Убар Р.Р. Универсальный подход к автоматизации проектирования тестов для широкого класса дискретных объектов. См. наст. сб., с. 75.

2. Виноград Т. Программа, понимающая естественный язык. - М.: Мир, 1976. 281 с.

3. Тыугу Э.Х. Концептуальное программирование. - М.: Наука, 1984. 255 с.

4. Woods W.A. Transition networks for natural language analysis. - Comm. ACM, 1970, 13, p. 591-606.

5. Akers S.B. Binary decision diagrams. - IEEE Trans. on Computer, June 1978, vol. c-27, p. 509-516.

6. Убар Р.Р. Обобщенная модель альтернативных графов для синтеза тестов цифровых систем. - Тр. Таллинск. политехн. ин-та, 1983, № 550, с. 97-109.

V. Alango, T. Kont

Preprocessor for an Automatic Test-generation System

Abstract

In this paper the preprocessor for an automatic test-generation system is described. The theoretical principles and structure of preprocessor are given. This program package allows to describe digital circuits at the conceptual level by the means of a problem-oriented subset of natural language. The preprocessor is implemented on a CM-4 minicomputer.

№ 626

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.32

К. Григорьева, Н. Эйнасто

ОБ АВТОМАТИЗАЦИИ ПРОЦЕДУР СИНТЕЗА ТЕСТОВ ДЛЯ ДИСКРЕТНЫХ СИСТЕМ НА МОДЕЛИ АЛЬТЕРНАТИВНЫХ ГРАФОВ

Проведенный анализ основных направлений построения тестов для дискретных систем (ДС) показал, что существующие в настоящее время методы основываются на разных языках описания объекта и на различных моделях неисправностей. Отсутствие единой математической модели объекта диагностирования и соответствующей ей модели неисправностей затрудняет создание универсальной системы автоматизированного проектирования тестов, пригодной для широкого класса ДС. Ниже обсуждаются возможности применения модели альтернативных графов [1, 2] для восполнения этого пробела.

Рассмотрим обобщенные альтернативные графы (АГ), определенные в [2], и выделим в них подкласс простых двоичных АГ, где все вершины отмечены только двоичными переменными. На такой модели основывается система автоматизированного проектирования тестов, описываемая в [3, 4] и разработанная совместно на кафедре ЭВМ ТПИ и в СКБ ВТ Института кибернетики АН ЭССР. Система использовалась до настоящего времени для проектирования тестов для нерегулярных дискретных объектов средней размерности. Рассмотрим теперь некоторые возможности применения этой системы для автоматизации процессов построения тестов для сложных дискретных систем высокой размерности, в частности, для микропроцессорных БИС и систем.

Разобьем диагностируемую ДС на две части – на операционный автомат (ОА) и управляющий автомат (УА). Процесс построения тестов проводим в два этапа: на первом этапе строятся тесты для ОА, на втором этапе – для УА.

Тесты для ОА могут быть построены автоматически при помощи рассматриваемой системы [3, 4] по его внутренней

структуре, если последняя известна. В противном случае тесты могут быть генерированы путем полного перебора (т.н. "тривиальный тест") или случайным образом [5].

Тесты для УА могут быть генерированы этой же системой на основе АГ-модели, полученной путем преобразования обобщенной АГ-модели [2] в простые логические АГ, в которых терминальные вершины обобщенных АГ, взвешенные функциями ОА $F_i(Z_{F,i})$, заменяются простыми двоичными АГ [I] $G_i=(M_i,\Gamma_i)$, где

$$M_i=\{m_1,\ldots,m_n,m_{n+1},m^*\},$$
$$\Gamma_i^{-1}(m_1)=\Gamma_i(m_{n+1})=\Gamma_i(m^*)=\phi, \qquad (I)$$
$$\forall_j=\overline{1,n}:\Gamma_i(m_j,1)=m_{j+1},\ \Gamma_i(m_j,0)=m^*,$$

причем вершина m_{n+1} взвешена вспомогательной булевой переменной f_i, представляющей функцию $F_i(f_i=1(0)$, если функция активизирована, $f_i=0(1)$, если функция потенциально активизирована, т.е. реализуется в присутствии проверяемой неисправности), вершины $m_j, j=\overline{1,n}$, взвешены двоичными переменными $z_K\in Z_{F,i}$, $|Z_{F,i}|=n$, представляющими аргументы (операнды) функции F_i ($z_K=1$, если операнд участвует в активизируемой (или потенциально активизируемой в случае неисправности) операции и требуется установка его значения), вершина m^* взвешена неопределенной константой (при детерминированном решении задачи построения тестов вход в эту вершину запрещен).

В свете вышесказанного множество всех переменных АГ-модели разбивается на следующие подмножества:

$$Z_{АГ}=Z_y\cup Z_F\cup Z_0, \qquad (2)$$

где Z_y – множество двоичных переменных, описывающих поведение УА,

Z_F и Z_0 – множества двоичных переменных, организующих связь между моделями УА и ОА и представляющих, соответственно, активизируемые операции и устанавливаемые операнды.

На такой АГ-модели тест получается в общем случае путем фиксирования значений 0 или I некоторых подмножеств переменных $Z'_y\subset Z_y$ и $Z'_F\subset Z_F$ и значения I некоторого подмножества переменных $Z'_0\subset Z_0$. Для всех $z_K\in Z'_0$ происходит обращение

к соответствующему графу G_K с целью установки значения $z_K=1$. При построении теста активизируется один основной путь с установлением значения $e \in \{0,1\}$ для некоторой $f_K \in Z'_F$ и соответствующие дополнительные пути с установлением значения $\forall f_i \in Z'_F \setminus f_K : f_i = \bar{e}$, соответственно, для всех проверяемых данным тестом неисправностей. Полученный тестовый набор следует интерпретировать как задачу дополнительного решения следующего множества неравенств:

$$\forall f_i \in Z'_F \setminus f_K : F_i(Z_{F,i}) \neq F_K(Z_{F,K}), \qquad (3)$$

решение которых обеспечивает получение конкретных значений для устанавливаемых операндов, представленных переменными $z \in Z'_0$.

В качестве примера на рис. I приведена АГ-модель, описывающая поведение регистровой пары BC микропроцессора К580ИК80. Здесь все переменные из множеств $Z_y=\{I_0,I_1,\ldots,I_7\}$, $Z_F=\{F_0,F_1,\ldots,F_{11}\}$ и $Z_0=\{BC,IN,IN2,\ldots\}$ - логические переменные. При этом I_i представляют разряды формата команды микропроцессора, а интерпретация переменных $F_i \in Z_F$ и $z_j \in Z_0$ следующая: при $F_i \in \{0,1\}$ в объекте активизирована функция, соответствующая переменной F_i, при неопределенном значении $F_i = x$ эта функция не активизирована, при $z_j=1$ требуется установка конкретного значения для операнда, представленного переменной z_j (при $z_j=0$ значение этого операнда безразлично). Под F_0 понимаем формально функцию, не изменяющую значение регистровой пары BC, для остальных F_i выполняемая функция обозначена мнемоникой системы команд микропроцессора на входе соответствующих вершин на рис. I и дешифрирована конкретно в табл. I. Жирными стрелками на рис. I изображены активизируемые пути на АГ при построении теста для проверки неисправности $I_6 \equiv 0$. Этим путям соответствует тестовый набор: $I_0=0, BC'=1, I_5=I_4=I_7=0, I_6=R=I_3=F_2=1$ (основной путь), $I_3=I_2=I_1=IN2=1, F_4=0$ (дополнительный путь). Из $BC'=1$ и $R=A'=1$ следуют установочные задачи, а из $F_2=1$ и $F_4=0$ следует неравенство $F_2 \neq F_4$ или, согласно соответствующим выполняемым функциям MVI C и MOV C,A, неравенство операндов $IN2 \neq A'$, где IN2 - вводимый при выполнении команды MVI C байт данных. Через штрих при переменной обозначена принадлежность значения переменной к предыдущему моменту времени (в данном случае - значение переменной после выполнения предыдущей команды).

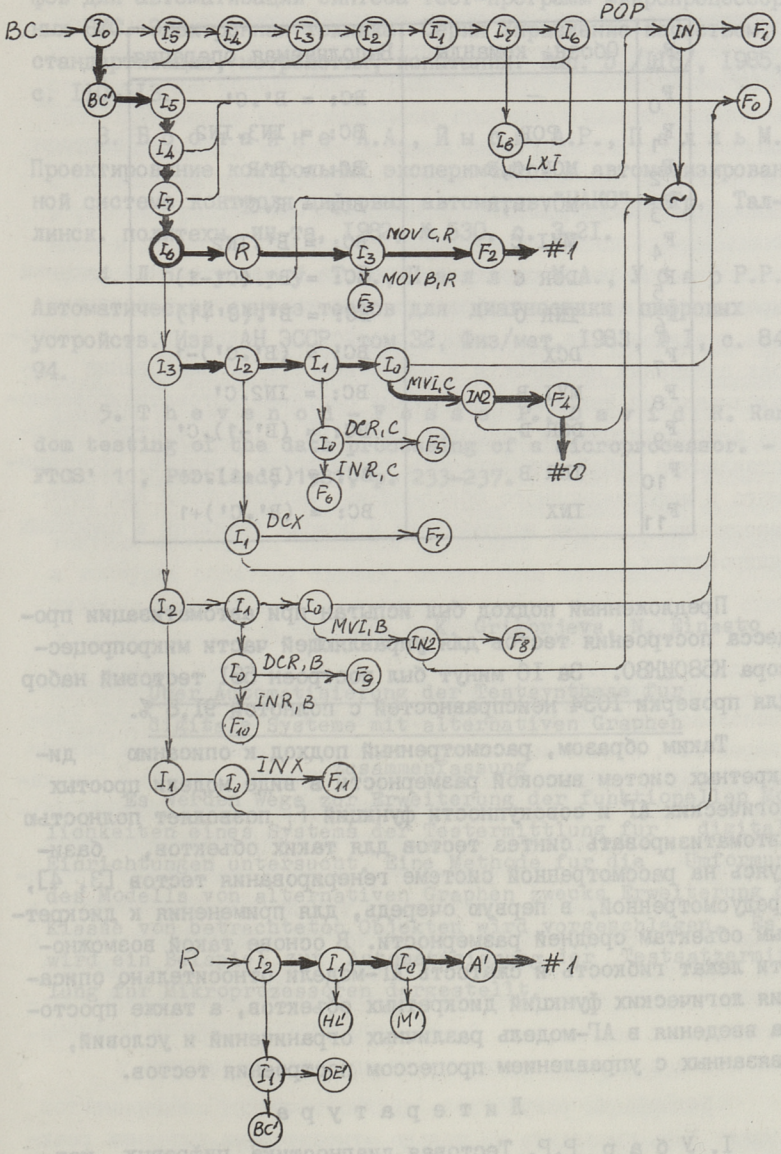

BC
POP
IN
LXI
MOV C,R
MOV B,R
#1
MVI,C
DCR,C
INR,C
#0
DCX
MVI,B
DCR,B
INR,B
INX
R
A'
HL'
M'
DE'
BC'

Рис. 1.

Таблица I

F_i	Обозн. команды	Выполняемая операция
F_0	—	BC: = B'.C'
F_1	POP	BC: = IN3.IN2
F_2	MOV C,R	BC: = B'.R
F_3	MOV B,R	BC: = R.C'
F_4	MVI C	BC: = B'.IN2
F_5	DCR C	BC: = B'.(C'-1)
F_6	INR C	BC: = B'.(C'+1)
F_7	DCX	BC: = (B'.C')-1
F_8	MVI B	BC: = IN2.C'
F_9	DCR B	BC: = (B'-1).C'
F_{10}	INR B	BC: = (B'+1).C'
F_{11}	INX	BC: = (B'.C')+1

Предложенный подход был испытан при автоматизации процесса построения тестов для управляющей части микропроцессора К580ИК80. За I6 минут был построен 58I тестовый набор для проверки I034 неисправностей с полнотой 9I,8 %.

Таким образом, рассмотренный подход к описанию дискретных систем высокой размерности в виде модели простых логических АГ и совокупности функций F_i позволяет полностью автоматизировать синтез тестов для таких объектов, базируясь на рассмотренной системе генерирования тестов [3, 4], предусмотренной, в первую очередь, для применения к дискретным объектам средней размерности. В основе такой возможности лежат гибкость и сжатость АГ-модели относительно описания логических функций дискретных объектов, а также простота введения в АГ-модель различных ограничений и условий, связанных с управлением процессом построения тестов.

Литература

I. Убар Р.Р. Тестовая диагностика цифровых устройств. I часть. Таллин, ТПИ, I980. II4 с.

2. У б а р Р.Р. Применение модели альтернативных графов для автоматизации синтеза тест-программ микропроцессорных БИС. Электронная техника. Серия Управление качеством, стандартизация, метрология, испытания. Вып. 5 /II6/, I985, с. II0-II3.

3. В о о л а й н е А.А., Й ы г и А.Р., П а л л ь М.А. Проектирование контрольных экспериментов в автоматизированной системе контроля цифровых автоматов "НАКС" - Тр. Таллинск. политехн. ин-та, I982, № 530, с. 3-2I.

4. Л о х у а р у Т.В., П а л л ь М.А., У б а р Р.Р. Автоматический синтез тестов для диагностики цифровых устройств. Изв. АН ЭССР, том 32, Физ/мат, I983, № I, с. 84-94.

5. T h e v e n o d - F o s s e P., D a v i d R. Random testing of the data processing of a microprocessor. - FTCS' 11, Portland, 1981, p. 233-237.

K. Grigorjeva, N. Einasto

Über Automatisierung der Testsynthese für digitale Systeme mit alternativen Graphen

Zusammenfassung

Es werden Wege zur Erweiterung der funktionellen Möglichkeiten eines Systems der Testermittlung für digitale Einrichtungen untersucht. Eine Methode für die Umformung des Modells von alternativen Graphen zwecks Erweiterung der Klasse von betrachteten Objekten wird vorgeschlagen. Es wird ein Beispiel zur Automatisierung der Testsatzermittlung für Mikroprozessoren dargestellt.

№ 626

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 681.32

Т.А. Эвартсон

ЛОКАЛИЗАЦИЯ ДЕФЕКТОВ В ЦИФРОВЫХ СХЕМАХ

Широкое распространение средств вычислительной техники предъявляет высокие требования к их качеству и эффективности использования в народном хозяйстве. Важная роль в вопросах повышения надежности и эффективности использования вычислительной техники принадлежит системам автоматизированного диагностирования цифровых схем (ЦС). Такие системы, представляющие собой совокупность технических средств, программного и информационного обеспечения, используются для автоматизации проверки исправности и поиска дефектов в цифровых устройствах.

I. Общие принципы поиска дефектов в ЦС

Традиционным подходом к диагностированию ЦС является использование таблиц неисправностей и диагностических словарей [I]. Применение такого подхода ограничивается резким увеличением размерности цифровых объектов, большими списками возможных неисправностей и связанным с этим большим объемом диагностической информации, затрудняющим реализацию методов автоматизации диагностирования технического состояния объекта. Основные недостатки таблиц неисправностей и диагностических словарей, кроме их громоздкости, связаны с их недостаточностью при локализации дефектов: учитываемый в них класс дефектов обычно ограничен, а учет вообще представляется нереальным. Диагностические словари не применимы также в случае, когда тесты являются неполными.

Дешифрация результатов диагностических экспериментов может быть проведена не только относительно выходных реакций объекта, но и с учетом состояний объекта в его внутренних контрольных точках. В связи с этим широкое применение нашли методы внутрисистемного испытания [2] и методы

поиска дефектов при помощи логических пробников [3]. Наиболее мощными средствами локализации дефектов в ЦС являются приспособления типа "игловых подушек" для одновременного измерения всех внутренних точек проверяемой платы. Однако, они экономны только при серийном выпуске проверяемых объектов. Применение многоканальных пробников для измерения одновременно всех контактов одной интегральной схемы связано с появлением возмущений в работе объекта при проверке его в рабочем режиме на высоких частотах. Наиболее универсальными и дешевыми при поиске дефектов являются одноканальные пробники, которые, однако, без применения средств оптимизации требуют трудоемких и длинных процедур диагностирования.

На практике используются многократные логические пробники. При использовании многотактных пробников исключаются длинные процедуры обратного прослеживания логических сигналов во времени, а применение сигнатур обеспечивает резкое сжатие диагностической информации. Недостатками известных методов поиска дефектов, основанных на применении многотактных пробников, являются трудность локализации дефектов в контурах обратных связей, отсутствие возможностей оптимизации маршрутов прослеживания ложных сигналов по схеме объекта, потеря диагностической информации о тактах появления ложных сигналов и др. Основным недостатком однотактных пробников является трудоемкость поиска дефектов в динных сегментах теста.

В случае длинных тестовых последовательностей представляется невозможным предварительное вычисление всей эталонной информации без принятия мер для сжатия ее. С другой стороны, сжатие эталона в виде сигнатур не способствует оптимизации маршрутов поиска дефектов. Это вызывает необходимость разработки новых методов анализа длинных тестов, сочетающих традиционные логические методы моделирования со специальной обработкой числа импульсов в регулярных частях длинных последовательностей [4].

2. Генерирование диагностической информации для поиска дефектов с одноканальным пробником

Предлагается метод организации диагностических экспериментов, по которому диагностическая информация - эталонные значения сигналов во внутренних контрольных точках -

генерируется частично или полностью в процессе самого эксперимента. В результате этого достигается резкое снижение объема памяти, требуемого для хранения эталонной информации. Предлагаемый метод основывается на селективном моделировании объекта на непрошедшем наборе (или предыдущих наборах) теста, требующего обработки лишь той части объекта, в которой действует неисправность и распространяются ложные сигналы. Селективное моделирование базируется на модели альтернативных графов (АГ) [5], обеспечивающая простоту алгоритмов обработки в ЭВМ и содержащая информацию о структуре и функции ЦС.

Рассмотрим некоторый модуль ЦС, реализующий функцию $z_k = f(Z_k)$, где $Z_k \subset Z$ – множество входных переменных модуля, а Z – множество всех переменных объекта. При ложном значении z_k возникает вопрос, какие из входов $z_i \in Z_k$ следует проверить при помощи пробника. Очевидно, проверить следует только те $z_i \in Z_k$, изменения значений которых может привести к изменению значения z_k.

При обнаружении ложного сигнала на одном из входов $z_i \in Z_k$ происходит переход к проверке соответствующего модуля f_i с функцией $z_i = f_i(Z_i)$ (если $z_i \in Z_{вх}$, где $Z_{вх} \subset Z$ – множество входных переменных объекта, то неисправность считается локализованной на входе z_i). При отсутствии ложных сигналов на входах $z_i \in Z'_k$ неисправным считается сам модуль f_k.

Определение необходимости проверки значений $z_i \in Z_k$ возможно на основе булевых дифференциальных уравнений, однако это связано с громоздкими вычислениями.

Рассмотрим ниже более эффективный способ отсеивания бесперспективных проверок, основанный на применении модели АГ.

Представим каждую функцию $f_k \in F$ в виде графа

$$G_k = \{M_k, \Gamma_{k,e}\}, \; e \in \{0,1\},$$

где M_k – конечное непустое множество вершин, а Γ_k – отображение M_k в M_k. Через $\Gamma_k(m,e)$ обозначим последователя вершины $m \in M_k$ в направлении e. Из $e \in \{0,1\}$ следует, что рассматриваемые графы являются альтернативными (из каждой вершины выходит не более двух дуг).

Каждой вершине $m \in M_K$ в графе G_K сопоставлена весовая функция

$$e(m) = \beta(m) \oplus z(m)^{t-\tau(m)},$$

где β, τ – некоторые признаки, принадлежащие вершине m. Здесь $\beta(m)$ – признак инвертирования весовой переменной $z(m)$, а $\tau(m)$ – признак для указания запаздывания на один такт в случае последовательных схем ($\tau(m)=1$, если $z(m) \in Z_n$, где $Z_n \subset Z$ – множество переменных состояния объекта).

Введем теперь оператор $G_K(m^H,t)$ движения в графе G_K, начинающегося с начальной вершины m^H и выходящего из каждой вершины m_i к вершине $m_j = \Gamma_K(m_i, e(m_i))$, до выхода из графа в некоторой терминальной вершине m_s, где $\Gamma_K(m_s, e(m_s)) = \Phi$, определяющей значение оператора в следующем виде:

$$G_K(m^H,t) = e(m_s).$$

Процесс выполнения оператора $G_K(m^H,t)$ можно рассматривать как моделирование заданной данным графом функции.

Селективное моделирование ЦС базируется на применении оператора $G_0(m_0^H,t^*)$ для графа G_0, где $m_0^H \in M_0$, а переменная $z_0 \in Z_{вых}$ соответствует выходу ЦС с ложным сигналом в такте t^*. При выполнении оператора $G_0(m_0^H,t^*)$ образуется некоторый путь $\ell_0(t^*) \subseteq M_K$, образованный из пройденных в графе G_0 вершин. Для вершин $m \in \ell_0(t)$, где $z(m) \notin Z_{вх}$, требуется обращение к другому оператору $G(z(m), t^* - \tau(m))$ с целью вычисления значения $z(m)$ в соответствующем графе, иначе не может быть продолжено движение в исходном графе G_0.

Пусть $M'_K \subseteq M_K$ – множество пройденных при моделировании вершин графа G_K, а M' – множество всех пройденных вершин в системе АГ. Результаты селективного моделирования можно представить в виде дерева Q, состоящего из вершин $m \in M'$. Корень дерева Q определяется формально как вершина m_0, с которой сопоставлены граф G_0 и соответствующая подсистема ЦС с выходом $z_0 \in Z_{вых}$.

Для любой невисячей вершины $m \in M'$ дерева Q, взвешенной некоторой переменной $z_K \in Z \setminus Z_{вх}$, где $Z_{вх} \subset Z$ – множество входных переменных ЦС, последователями являются вершины $m \in M'_K$, пройденные при селективном моделировании в графе G_K с целью вычисления значения z_K. Висячими вершинами де-

рева Q служат те $m \in M'$, которые взвешены входными переменными ЦС $z(m) \subset Z_{bx}$. Такое дерево будем называть диагностическим деревом (ДД) и оно является основой при организации процедур поиска неисправностей.

Вершины $m \in M'$, для которых $z(m) \in Z^*$ представляют контрольные точки ЦС. Здесь $Z^* \subset Z$ – множество переменных, представляющих сигналы, доступные пробнику. Локализацию дефектов в ЦС можно теперь провести путем опроса пробником реальных значений $z(m) \in Z^* (m \in M')$ и сравнения их с соответствующими эталонными значениями.

Итак, в результате селективного моделирования находятся не только эталонные значения для потенциальных контрольных точек, но и образуется стратегия поиска дефектов в виде ДД. При этом в ДД включаются не все входы потенциального неисправного элемента, а лишь те, которые ответственны за значение сигнала на выходе элемента.

Далее покажем, что не все $z(m), m \in M'$ требуют измерения пробником. Обозначим через $\ell_K(m, D)$ путь в АГ G_K, активизированный текущим набором теста, начинающийся в вершине и выходящий из графа в направлении $D \in \{0,1\}$.

Пусть неисправность требуется искать в такте t по некоторому пути $\ell_K^t(m^H, D)$ в графе G_K.

Введем следующее разбиение:

$$M_K = M_K^A \cup M_K^B .$$

Определение I. Определим множество $M_K^A \subseteq M_K$ такое, что $m \in M_K^A$, если не существует такого набора значение переменных Z_K, порождающих путь $\ell_K(m^H, D)$, который проходил бы через вершину m при значении $e(m) = \bar{D}$ так, чтобы изменение $e(m)$ не порождало другого пути $\ell_K(m, \bar{D})$.

Теорема I. Если при моделировании набора Z_K в такте в графе G_K образуется путь $\ell_K^t(m^H, D)$, проходящий через множество вершин M_K', то источником ложного сигнала не могут быть вершины $m \in M_K' \cap M_K^A$, для которых $e(m) = \bar{D}$.

Из теоремы I следует возможность отсеивания бесперспективных проверок с пробником. Если на выходе некоторого модуля ЦС с функцией $z_K = f_K(Z_K)$ и АГ G_K обнаружен ложный сигнал, то в множество требующих проверок контрольных точек на выходе модуля достаточно включить весовые перемен-

ные $z(m)$ при вершинах $m \in \ell_K^t(m_K^H, D) \cap M_K^A$, для которых $e(m) = D$ (т.е. направление выхода из вершины должно совпадать с выходом из графа), а также при всех вершинах $m \in \ell_K^t(m_K^H, D)$, для которых $m \notin M_K^A$.

Здесь множество $M_K^B \subseteq M_K$ такое, что $m \in M_K^B$, если существует набор Z_K, порождающий путь $\ell_K(m^H, D)$, который проходит через вершину m при значении $e(m) = \bar{D}$, и изменение значения $e(m)$ может порождать некоторый путь $\ell_K(m, \bar{D})$.

Очевидно, что если при моделировании набора Z_K в такте t в графе G_K образуется путь $\ell_K^t(m^H, D)$, проходящий через множество вершин M'_K, то источником ложного сигнала всегда могут служить вершины $m \in M'_K \cap M_K^B$.

Можно доказать, что все вершины графов функции И, ИЛИ и НЕ принадлежат множеству M^A, все вершины графов, построенных из них методом суперпозиции также принадлежат множеству M^A.

Определение 2. Определим множество $M_K^C \subseteq M_K$, такое что $m \in M_K^C$ для данного набора Z_K, порождающего путь $\ell_K(m^H, D)$, если при изменении значения $e(m)$ не может порождаться некоторый путь $\ell_K(m, \bar{D})$.

Теорема 2. Если при моделировании набора Z_K в такте t в графе G_K образуется путь $\ell_K^t(m_K^H, t)$, проходящий через множество вершин M'_K, то источником ложного сигнала при данном наборе Z_K не могут служить вершины $m \in M'_K \cap M_K^C$.

На основе рассмотренных выше свойств АГ определяется сокращенное множество контрольных точек M_K^{KT}, требующих проверки с пробником $M_K^{KT} = ((M_K^B \cup (M_K^A \setminus M_K^{A'})) \setminus M_K^C) \cap M'_K$, где $M_K^{A'} = M'_K \cap M_K^A$, для которых $e(m) = \bar{D}$.

3. Оптимизация процесса поиска дефектов в ЦУ

Можно определить следующие подходы к оптимизации поиска дефектов: локальная оптимизация и глобальная оптимизация.

Под локальной оптимизацией поиска понимаем решение задачи, в каком порядке проверять входы модуля, если на выходе обнаружен ложный сигнал. Приоритет, очевидно, следует отдавать точке, где вероятность обнаружения ложного сигнала максимальна.

Здесь и в дальнейшем предполагается, что из первоначального ДД, полученного селективным моделированием уже исключены все "бесперспективные" варианты поиска дефектов.

Рассмотрим в ДД поддерево $Q(m_k)$ с корнем m_k. При обнаружении ложного сигнала на входе $z(m)$ модуля, где m – последователь вершины m_k в дереве $Q(m_k)$, число возможных исходов поиска при его продолжении определяется числом вершин в поддереве $Q(m)$. Пусть $p(m_i)$ априорная вероятность появления неисправностей, местонахождение которых определено точкой $z(m_i)$. Тогда вероятность обнаружения ложного сигнала на входе $z(m)$ будет определяться как

$$P(m)=\frac{p(m)}{\sum\limits_{m_j \in Q(m_k)\setminus m_k} p(m_j)+p(m_k)}.$$

Предполагая равновероятные неисправности неисправностей можно использовать более простую формулу

$$P(m)=\frac{|Q(m)|}{\sum\limits_{m_j \in Q(m_k)\setminus m_k} |Q(m_j)|+1}.$$

С целью минимизации среднего числа проверок на входе модуля f_k необходимо теперь упорядочить контрольные точки в порядке убывания $P(m)$, а на каждом шаге их оставшихся непроверенных точек $m \in M_k^{кт}$ выбрать ту, для которой $P(m)$ имеет максимальное значение.

Под глобальной оптимизацией поиска понимаем решение задачи выбора контрольной точки по всему диагностическому дереву, в котором вероятность обнаружения ложного сигнала максимальна. Глобальная оптимизация более трудоемка, чем локальная оптимизация, но она может дать лишь точное решение минимизации средней длины процесса поиска задачи, в то время как локальная оптимизация всегда дает лишь приближенное значение.

Рассмотрим полное диагностическое дерево Q, полученное после отсеивания всех вершин m (а также соответствующих поддеревьев $Q(m)$, относящихся к бесперспективным проверкам). Каждую вершину $m \in Q$ можно характеризовать вероятностью $P(m)$ появления ложного сигнала в контрольной точке $z(m)$ при условии, что на выходе схемы наблюдается ложный сигнал:

$$P(m) = \frac{\sum_{m_i \in Q(m)} p(m_i)}{\sum_{m_j \in Q} p(m_j)}.$$

При предположении равных вероятностей $p(m_i), m_i \in Q$ может быть использована формула

$$P(m) = \frac{|Q(m)|}{|Q|}.$$

Согласно методике построения кода Шэннона-Фано, требуется в первую очередь провести проверку в точке m, носящей максимальное количество информации, т.е. где $P(m) = 0{,}5$ или $P(m)$ наиболее близок к 0,5. В зависимости от результата проверки определяется следующая область поиска: при отрицательном исходе за основу берется дерево $Q(m)$ при положительном исходе - дерево $Q \setminus Q(m)$. На полученном дереве вычисляются новые значения $P(m)$ и процедура упорядочения проверок повторяется.

Можно выделить и некоторые промежуточные варианты, базирующиеся не иерархическом подходе к проверке, в частности, этапы грубой локализации и точной локализации дефекта.

Выделим три уровня, на которых может быть реализован иерархический подход к поиску дефектов:

- уровень дефектов;
- уровень связей;
- уровень модулей.

На уровне дефектов путем грубой локализации устанавливается контрольная точка с ложным сигналом как можно ближе к физическому местонахождению неисправности, затем путем точной локализации уточняется причина ложного сигнала. Спецификация отдельных неисправностей $r_{ki} \in R_k$, способных непосредственно влиять на значение z_k, задается в виде множества дополнительных логических условий локальной активизации неисправностей

$$W_{ki}(Z) = 1, \quad i : r_{ki} \in R_k.$$

Иерархический подход к локализации неисправностей на уровне связей заключается в следующем. На этапе грубой локализации сигналы на связях между интегральными схемами проверяются лишь в одной точке (на входах ИС). При обнаружении ложного сигнала на входе z_k переходим сразу к про-

верке входов элемента f_k, не проверяя его выход и т.д. При правильном сигнале на входах элемента f_k начинается этап точной локализации дефекта. Дефектом может быть как элемент, так и связь z_k, соединяющая элементы f_k и f_j. Для уточнения вопроса требуется дополнительная проверка непосредственно выхода z_k элемента f_k или его окрестности.

На уровне модулей этап грубой локализации осуществляется путем проверки сигналов на входах и выходах укрупненных подсхем до обнаружения только правильных сигналов на входах некоторой подсхемы. Затем начинается точная локализация путем прослеживания ложных сигналов на внутренних точках неисправной подсхемы.

Предложенные принципы реализованы в СКБ ВТ АН ЭССР в виде комплексов программ, включенных в состав двух автоматизированных систем контроля, осуществляющих полуавтоматическую локализацию неисправностей в диалоговом режиме с оператором. Универсальный программно-управляемый стенд проверки логических ТЭЗ-ов ЕС ЭВМ построен на базе ЭВМ СМ-3 и усовершенствованный вариант стенда построен на базе ЭВМ СМ-4.

Литература

I. Основы технической диагностики / Под ред. П.П. Пархоменко. - М.: Энергия, 1976, с. 464.

2. Mastrocola A. In-circuit test techniques applied to complex digital assemblies. - IEEE Test Conference, 1981, p. 124-131.

3. Zobniw L.M. Real time diagnosis using single pin probe. - 12th Des. Automat. Conf. Proc., Boston, Mass., 1975, p. 268-286.

4. Убар Р., Эвартсон Т. О моделировании длинных входных последовательностей в дискретных устройствах, содержащих счетные структуры. - Тр. Таллинск. политехн. ин-та, 1985, № 601, с. 61-74.

5. Убар Р.Р. Описание цифровых устройств моделью альтернативных графов. - Тр. Таллинск. политехн. ин-та, 1979, № 474, с. 11-33.

T. Evartson

Fehlerlokalisierung in digitalen Schaltungen

Zusammenfassung

Die Fragen der Synthese der diagnostischen Information für die Fehlersuche mit Prüfstift werden betrachtet. Es werden Sätze zwecks Minimierung der Anzahl von Prüfschritten formuliert. Verschiedene Methoden zur Optimierung der Prüfprozesse werden diskutiert.

№ 626
TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА
УДК 621.374.2

В.А. Герасимчук

ПРЕЦИЗИОННЫЕ КОМПАРАТОРЫ АНАЛОГОВЫХ СИГНАЛОВ

Компараторы занимают промежуточное положение между аналоговыми и цифровыми функциональными узлами, являются, по существу, простейшими аналого-цифровыми преобразователями, находят применение в схемах быстродействующих аналого-цифровых преобразователей, в схемах автоматики и измерительной техники. В реальных жизненных ситуациях мы имеем дело с явлениями, процессами, характеризуемыми соответствующими величинами, представленными в аналоговой форме, поэтому автоматизированные системы, построенные на основе цифровой вычислительной техники, должны включать в себя аналого-цифровые и цифроаналоговые преобразователи для реализации связи с реальными объектами. На долю этих преобразователей падает большая часть затрат труда как на наиболее сложные узлы, определяющие эффективность и качество всей системы в целом. Поэтому задача преобразования аналогового сигнала в цифровой, хотя и кажется достаточно традиционной, зачастую все же требует для своего решения некоторых затрат времени и энергии.

Промышленностью выпускаются интегральные компараторы, выполняющие необходимое преобразование, однако для увеличения их функциональной надежности, т.е. увеличения помехоустойчивости, получения крутых фронтов при медленных изменениях входного сигнала, устранения "дребезга" приходится вводить в схему положительную обратную связь (рис. I а) [I]. Это приводит к появлению у компаратора гистерезиса, т.е. к разнице значений напряжений порогов включения и выключения схемы. Из приведенных на рис. I б временных диаграмм видно изменение порогового напряжения $U_п$ компаратора. Наличие гистерезиса приводит к дополнительной погрешности времени переключения компаратора (Δt), что является серьезным недостатком подобного решения. При наличии на входе компара-

тора шумов погрешность времени переключения может быть представлена в виде:

$$\Delta t = \Delta t_0 + \Delta t_1 ,$$

где Δt_0 - систематическая составляющая;

Δt_1 - случайная составляющая.

Величина случайной составляющей определяется отношением шум/сигнал входного сигнала. При некоррелированном с входным сигналом шуме математическое ожидание этой составляющей равно нулю. Предварительной фильтрацией можно уменьшить ее величину. Кроме того, следует учитывать, что наличие шума может вызвать в момент переключения пачку импульсов. Это особенно опасно для цифровых схем, если выходной сигнал компаратора используется в качестве счетного импульса в этих схемах.

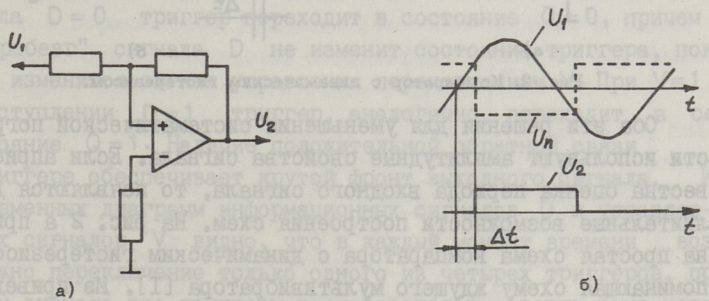


Рис. 1. Регенераторный компаратор.

Величина систематической составляющей зависит от напряжения гистерезиса $\Delta U_п$ и уровня входного сигнала, и очевидно определяется отношением $\Delta U_п / U_1$. Систематическая составляющая погрешности времени переключения носит мультипликативный характер, что делает малоэффективной аддитивную коррекцию этой погрешности. Известны решения, направленные на уменьшение этой составляющей погрешности за счет уменьшения влияния напряжения гистерезиса. В [2] предлагается изменять коэффициент передачи усилителя, стоящего перед компаратором, в зависимости от уровня сигнала на выходе этого усилителя, причем коэффициент передачи усилителя максимален, когда сигнал проходит нулевое значение. Это приводит к уменьшению систематической составляющей по-

грешности, но она все же остается. В [3] предлагается интересное решение, сущность которого заключается в компенсации напряжения гистерезиса основного компаратора при помощи дополнительного компаратора, сигнал которого суммируется с входным сигналом на входе основного компаратора. Как и всякая компенсация, эта компенсация имеет ограничения по точности, что и ограничивает точность всей схемы в целом.

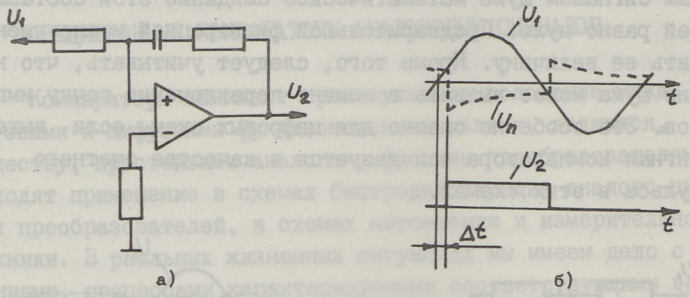


Рис. 2. Компаратор с динамическим гистерезисом.

Оба эти решения для уменьшения систематической погрешности используют амплитудные свойства сигнала. Если априорно известна оценка периода входного сигнала, то появляются дополнительные возможности построения схем. На рис. 2 а приведена простая схема компаратора с динамическим гистерезисом, напоминающая схему ждущего мультивибратора [1]. Из приведенных на рис. 2 б временных диаграмм видно, как по мере разряда конденсатора, напряжение порога срабатывания компаратора приближается к нулю. Тем самым, систематическая составляющая погрешности времени переключения уменьшается при той же самой глубине положительной обратной связи в момент переключения. Величина напряжения порога непосредственно перед переключением очевидно определяется выражением

$$U_n = \Delta U_{вых} \cdot \beta \cdot e^{-\frac{T_u}{\tau}},$$

где $\Delta U_{вых}$ – перепад выходного напряжения компаратора;
β – глубина обратной связи;
T_u – длительность выходного импульса компаратора;
τ – постоянная времени цепи обратной связи;

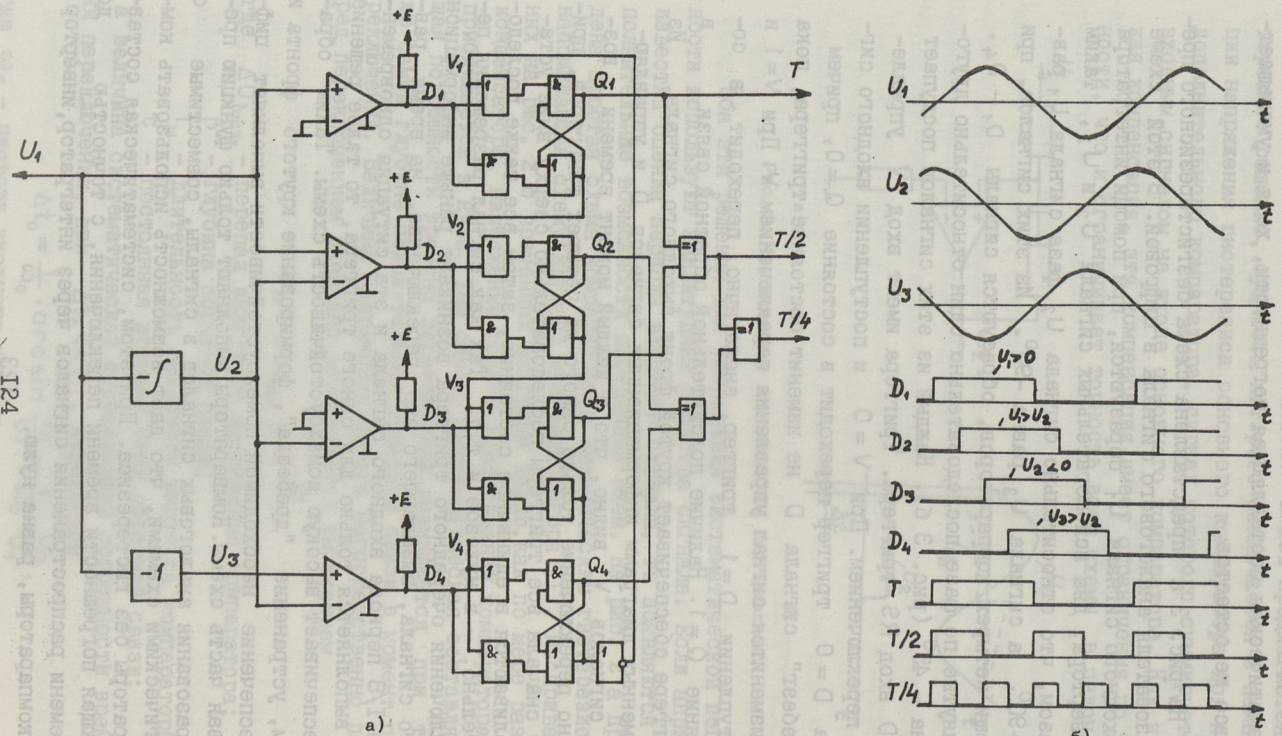
а величина систематической составляющей погрешности времени переключения, при $\Delta t_0 << T_u$, выражением

$$\Delta t_0 = \frac{1}{\omega_0} \cdot \arcsin \frac{U_n}{U_1} \approx \frac{1}{\omega_0} \cdot \frac{U_n}{U_1},$$

где ω_0 – частота входного сигнала.

Большим достоинством данного решения является простота, но систематическая составляющая погрешности, хотя и уменьшается, все же остается.

На рис. 3 а представлена схема безгистерезисного преобразователя аналогового сигнала в цифровой. В этой схеме из входного сигнала U_1 образуются, при помощи интегратора и инвертора, два вспомогательных сигнала U_2 и U_3, таким образом, что относительно сигнала U_2 фаза сигнала U_1 равна $+90^0$, а сигнала U_3 равна -90^0. Из этих сигналов, при помощи четырех компараторов, образуются сигналы $D_1 - D_4$, сдвинутые по фазе последовательно один относительно другого на $- 45^0$ (рис. 3 б). Каждый из этих сигналов поступает на D вход RS-триггера. Триггера имеют вход V управления переключением. При $V = 0$ и поступлении входного сигнала $D = 0$ триггер переходит в состояние $Q = 0$, причем "дребезг" сигнала D не изменит состояние триггера, пока не изменится сигнал управления переключением V. При $V = 1$ и поступлении $D = 1$ триггер, аналогично, переходит в состояние $Q = 1$. Наличие положительной обратной связи в триггере обеспечивает крутой фронт выходного сигнала. Из временных диаграмм информационных сигналов D и управляющих сигналов V видно, что в каждый момент времени возможно переключение только одного из четырех триггеров, причем сначала все триггеры последовательно сверху вниз устанавливаются в единичное состояние, затем все также последовательно, - в нулевое и т.д. Так как сигнал разрешения переключения очередного триггера возникает раньше информационного сигнала, вызывающего это переключение, на время, равное I/8 периода входного сигнала и эта ситуация одновременно выполняется только для одного триггера, то такое решение обеспечивает высокую помехоустойчивость схемы. Таким образом, устранение "дребезга", формирование крутого фронта и обеспечение необходимой помехоустойчивости выполняет цифровая часть схемы. Компараторы выполняют только функцию преобразования аналоговых сигналов в сигналы, совместимые с логическими схемами, что дает возможность использовать компараторы без гистерезиса. При этом, систематическая составляющая погрешности времени переключения, с точностью до времени распространения сигналов через интегратор, инвертор и компараторы, равна нулю.



а) б)

Рис. 3. Безгистерезисный преобразователь.

Вместо интегратора можно применять дифференциатор или фазовращатель, при этом несколько изменится схема соединения сигналов $U_1 - U_3$ с компараторами. В зависимости от конкретного применения это может дать возможность получения выигрыша в соотношении шум/сигнал на входах компараторов. Применение интегратора, как правило, требует включения после него повторителя через дифференцирующую цепочку. Это необходимо для устранения напряжения смещения нуля интегратора. При помощи трех элементов "исключающее или" реализована схема, обеспечивающая кроме основного сигнала T еще два дополнительных сигнала T/2 и T/4 с соответствующими периодами, которые могут оказаться полезными при разработке конкретного устройства.

К недостаткам этой схемы следует отнести высокую сложность и ограниченный диапазон изменения периода входного сигнала. Однако последнее обстоятельство скорее определяет специфику применения этой схемы.

Таким образом, разделение функций преобразования аналогового сигнала в сигнал, совместимый по уровням с логическими схемами и обеспечение необходимой помехоустойчивости, а также крутизны фронтов цифрового сигнала дало возможность повысить точность и функциональную надежность преобразователя.

Литература

I. Шило В.Л. Линейные интегральные схемы в радиоэлектронной аппаратуре. - М.: Сов. радио, 1979.

2. Алексеенко А.Г., Коломбет Е.А., Стародуб Г.И. Применение прецизионных аналоговых микросхем. - М.: Радио и связь, 1985.

3. Олсен С. Усилитель с двойной связью устраняет гистерезис в компараторе. - Электроника, 1980, № 14.

V. Gerasimtchuk

Precision Comparators for Analog Signals

Abstract

Circuits of precision comparators for analog signals are considered. The error of circuits with hysteresis are analysed. Several techniques are proposed to reduce error comparators.

№ 626

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 65.012

В.П. Махитько, М.Г. Шендрик, Б.Г. Тамм

РЕГУЛИРОВАНИЕ ПРОИЗВОДСТВЕННЫХ ПРОЦЕССОВ ПО КОЭФФИЦИЕНТУ РИТМИЧНОСТИ

Повышение эффективности производства, совершенствование методов планирования, регулирования и прогнозирования является актуальной задачей при обеспечении устойчивой работы механообрабатывающего цеха в течение планового периода.

Оперативное управление является одной из составляющих автоматизированной системы управления цехом (АСУ цехом).

Для планирования работ цеху, в составе задач АСУ цехом, формируется месячный оперативно-календарный план (ОКП), исходной информацией для которого является номенклатурный план, подготовленный на межцеховом уровне.

Главной целью формирования ОКП является организация согласованного во времени и пространстве движения деталей, сборочных единиц в производстве.

Правильностью построения ОКП считается:

1) полное, комплектное и равномерное выполнение производственной программы при соблюдении директивных сроков выпуска товарной и валовой продукции,

2) наиболее рациональное использование оборудования, трудовых и материальных ресурсов.

Критерием ОКП, в котором синтезируются все факторы воздействия на ход и конечные результаты производства, является показатель равномерности или ритмичности работы цеха.

В свою очередь, неритмичная работа цеха, проявляющаяся в "штурмовщине", отрицательно сказывается на технико-экономических показателях цеха, социально-психологическом климате коллектива, неравномерной загрузке оборудования,

снижении производительности труда, нарушении технологической дисциплины, повышении брака и т.д.

Ритмическая работа цеха - это согласованный, взаимоувязанный труд (подразделений, служб, линейного персонала цеха), направленный на выполнение производственной программы предприятия на основе ОКП цеха [I].

Ритмичность выпуска означает повторение одинакового или планомерного нарастающего объема выпуска товарной и валовой продукции через равные (например, декада) промежутки времени.

Равномерность производства означает воспроизведение через те же периоды времени одинакового или равномерно нарастающих по трудоемкости работ, обеспечивающих выпуск товарной и валовой продукции в текущий и последующие плановые периоды [I] и обозначается следующей формулой:

$$K_{в} = 1 - \frac{\Sigma B_{н}}{\Sigma B_{п}},$$

где $\Sigma B_{н}$ - величина неравномерности плана по объему выпуска продукции в абсолютных единицах (натуральных, трудовых, стоимостных) в определенные периоды времени (дни, часы);

$\Sigma B_{п}$ - плановый объем выпуска продукции за анализируемый период в соответствующих единицах.

Коэффициент равномерности выпуска продукции или коэффициент равномерности работы цеха могут быть определены по формуле:

$$K_{в} = \frac{B_{ф}}{B_{с}},$$

где $B_{ф}$ - фактический объем производственной продукции (работы) в единицу времени (час, сутки, смену, декаду и т.п.);

$B_{с}$ - средний объем производимой продукции (работы) за данную единицу времени по плану (среднечасовой, среднесуточной и т.д.).

Ритмичность производственных процессов определяется соотношением объемов работ, выполняемых на отдельных (равных по продолжительности) интервалах одного временного цикла. Обычно для числового выражения ритмичности производственных процессов вводят коэффициент ритмичности, который вычисляют по формуле [2]:

$$K_p = 1 - \frac{\sum_{i=1}^{n} (B_{пi} - B_{фi})}{\sum_{i=1}^{n} B_{пi}},$$

где $B_{пi}$ – плановый выпуск товарной (валовой) продукции за определенный период;

$B_{фi}$ – фактический выпуск товарной (валовой) продукции за определенный период;

$i = \overline{1, n}$ – число периодов, за которое анализируется ритмичность.

Как правило, на практике показатели ритмичности выпуска товарной (валовой) продукции и равномерности производства определяются по следующей формуле:

$$K_p = K_б = \frac{O_{фi}}{O_{пj}},$$

где $O_{фi}$ – фактический объем по трудоемкости, выполненный в i-ю отчетную декаду;

$O_{пj}$ – объем работ, планируемый цеху на j-й месяц.

При расчете показателя ритмичности выпуска товарной продукции K_p в числителе учитываются только позиции, имеющиеся в объеме производственной программы цеха. При исчислении показателя равномерности $K_б$ в числителе указывается весь объем выполняемых работ.

Приведенные показатели необходимы для анализа и сравнения результатов работы участков и цеха в целом, для установления нарушения равномерности производства и выпуска товарной (валовой) продукции, а также для принятия решений по ликвидации отклонений производства.

Из двух рассматриваемых показателей ритмичности работы цеха определяющим является показатель равномерности производства. Цеха и участки, где процессы основаны на планомерно организуемом режиме равномерного производства, обеспечивают и соблюдение ритмичного выпуска товарной (валовой) продукции. При правильно организованном движении производства эти показатели должны находиться в соотношении $K_p \geqslant K_б$.

Во временном аспекте K_p может быть плановым, рассчитанным на этапе формирования ОКП, и фактическим, отражающим реальную динамику свершившихся производственных процессов. Именно сопоставление динамики запланированной и фактически

достигнутой ритмичности является предметом исследования при анализе производственных процессов.

Коэффициент ритмичности при практическом использовании дает возможность оценить динамику производственного процесса и использовать полученную информацию для обоснования решений, принимаемых в сфере оперативного управления цехом.

Однако на предприятии и в цехах использование коэффициента ритмичности K_p для анализа производства имеет следующие недостатки:

- планируемые цехам коэффициенты ритмичности по декадам устанавливаются на низком уровне без учета научно обоснованных методов, что заведомо затрудняет правильность оценки ритмической работы цеха при сравнении планового и фактического выполнения объема товарной (валовой) продукции. В течение декады, как правило, осуществляется только расчет K_p за прошедшие интервалы без оценки сложившейся производственной ситуации и принятия оперативного решения,

- в большинстве случаев K_p за отдельный интервал в отношении к отчетному периоду рассчитывается по фактическому выполнению объемов товарной (валовой) продукции, даже в случае невыполнения плана, что приводит к искажению картины динамики K_p за прошедший отчетный период и т.п.

Определим некоторые понятия, которыми будем пользоваться при описании процедур расчета плановых заданий, учета и регулирования хода производства при использовании коэффициента ритмичности.

При расчете ОКП на основании графика комплектности сборочных позиций в сборочных цехах устанавливается плановый коэффициент ритмичности K_p^n механообрабатывающему цеху. Величину K_p^n можно определить из следующего соотношения [2]:

$$K_p^n = \frac{\sum_{i=1}^{3} B_{\phi i}}{\sum_{i=1}^{3} B_{ni}},$$

где $B_{\phi i}$ - фактический выпуск валовой продукции;

B_{ni} - плановый объем выпуска валовой продукции;

$i=(1,2,3)$ - число периодов (декад), за которое анализируется ритмичность.

Планируемый коэффициент ритмичности на i-ю декаду:

$$K_p^i = \frac{B_\phi^i}{B_n^i}.$$

В ходе производства всегда имеются расхождения с плановым заданием (включение в план директивных заданий, отклонения производства и т.д.), поэтому появляется необходимость перерасчета плана на последующие периоды (декады) месяца.

Расчет прогнозируемых значений коэффициента и выпуска валовой продукции на последующие периоды определяется с учетом фактического выпуска продукции в предыдущем периоде:

$$B_{np}^{i+1} = B_\phi^i \cdot \frac{K_p^{i+1}}{K_p^i},$$

где B_{np}^{i+1} – значение прогноза выпуска валовой продукции на плановую декаду;

B_ϕ^i – фактический выпуск продукции в текущей i-й декаде;

K_p^{i+1} – коэффициент ритмичности выпуска продукции в плановой $(i+1)$-й декаде;

K_p^i – коэффициент ритмичности выпуска продукции в текущей i-й декаде.

При отклонениях в ходе производства расчет отклонений прогноза от плана определяется:

$$\Delta^i = B_n^i - B_{np}^i,$$

где Δ^i – отклонение прогноза от плана в i-й декаде;

B_n^i – плановый объем выпуска продукции в i-й декаде;

B_{np}^i – прогнозируемый объем выпуска продукции в i-й декаде.

При опережении плана будут выполняться следующие неравенства:

$$B_\phi^i \geq B_n^i \quad \text{и} \quad K_p^i \geq K_{np}^i,$$

а в случае отставания:

$$B_\phi^i < B_n^i, \quad K_p^i < K_{np}^i.$$

Если абсолютная величина Δ^i меньше некоторого значения $\Delta_{ру}$, т.е. при условии $|\Delta^i| < \Delta_{ру}$, то выполнение плана может быть обеспечено без его корректировки за счет внутренних резервов системы управления. Значение $\Delta_{ру}$ определяется на этапе опытной эксплуатации и может служить критерием устойчивости нерегулируемой системы. Если отклонения Δ^i ве-

лики $|\Delta^i| \geq \Delta_{ру}$, то для определения управляющих воздействий следует провести перерасчет коэффициента ритмичности и плана для оставшегося планируемого периода:

$$B_{nn}^{i+1} = B_n^{i+1} + \Delta^i, \quad K_p^{i+1} = \frac{B_\phi^i}{B_n^i},$$

где B_{nn}^{i+1} – плановый объем выпуска продукции, пересчитанный на $i+1$-ю декаду;

B_n^{i+1} – плановый объем выпуска продукции на $i+1$-ю декаду.

В зависимости от соотношения величин K_p^i, K_p^n, $K_{ру}$ можно рассмотреть три варианта:

$$K_p^i > K_{ру},$$
$$K_p^n \leq K_p^i \leq K_{ру},$$
$$K_p^i < K_p^n,$$

где $K_{ру}$ – коэффициент устойчивости нерегулируемой системы.

Первый вариант $K_p^i > K_{ру}$ свидетельствует о том, что в последующие декады запланированную работу выполнить невозможно. Для выполнения планового задания необходимо выделить дополнительные ресурсы (увеличение числа оборудования, сверхурочные работы и т.д.).

Второй вариант: $K_p^n \leq K_p \leq K_{ру}$ свидетельствует о том, что план может быть выполнен с определенным перенапряжением. Необходимость перерасчета плана определяется по результатам анализа величины отклонения Δ^i. При выполнении условия $\Delta^i \geq \Delta_{ру}$ автоматически принимается решение о перерасчете плана.

Третий вариант: $K_p^i < K_p^n$, в этом случае план будет выполнен, причем имеется возможность включения дополнительных работ в плановое задание.

С учетом вышеизложенного разработан алгоритм автоматизированного регулирования производства в цехе. Этот алгоритм является составной частью общего алгоритма расчета ОКП, в подсистеме "Месячное планирование, анализ производства" интегрированной АСУ цехом:

Шаг I: Формирование плановых заданий, K_p^n, B_n.

Шаг 2: Анализ выполнения плана с учетом директивных изменений плана, данных учета, отклонений производства и др.

Шаг 3: Расчет прогнозных значений K_p^i, B_{np}^i.

Шаг 4: Расчет отклонений прогноза от плана, Δ^i.

Шаг 5: Анализ K_p^i, K_p^n, $K_{ру}$.

Шаг 6: Анализ условия $K_p^i > K_{ру}$, при котором требуется выделение дополнительных ресурсов.

Шаг 7: Анализ условия $K_p^i < K_p^n$, при котором требуется включение в план дополнительной номенклатуры деталей последующих периодов.

Шаг 8: Анализ условия $K_p^n < K_p^i \leq K_{ру}$, при котором определяется прогнозируемое отклонение последующих декад.

Шаг 9: Перерасчет плана, B_n^{i+1}, K_p^{i+1}.

Шаг I0: Печать выходных форм документов.

Таким образом, в составе задач оперативного управления цехом реализуется алгоритм регулирования производства по коэффициенту ритмичности.

Литература

I. Петров В.А., Масленников А.Н. Программно-целевая организация производства и оперативного управления в условиях групповой технологии и ГАП.-Л.: Лениздат, 1984.

2. Смирницкий Е.К. Экономические показатели промышленности.-М.: Экономика, 1980.

V. Mahitko, M. Shendrik,
B. Tamm

Production Schedule Preparation in the Complex Automatic Control System within a Workshop

Abstract

Some formal methods of production schedule preparation for the complex automatic control system within a workshop are discussed. For simplification the axiomatic basis of hierarchical systems control is given, allowing the use of approximation methods. The methods applied to achieve the production scheduling enable us to determine the final state of the control system, being the major concern of production control within a workshop. The methods used in the algorithms have several merits over the traditional methods applied in the automatic control systems.

№ 626

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

УДК 65.012

В.П. Махитько, М.Г. Шендрик, Б.Г. Тамм

ФОРМИРОВАНИЕ ПРОИЗВОДСТВЕННОЙ ПРОГРАММЫ В КОМПЛЕКСНОЙ АВТОМАТИЗИРОВАННОЙ СИСТЕМЕ УПРАВЛЕНИЯ ЦЕХАМИ (КАСУ ЦЕХАМИ)

Процедура формирования производственной программы в КАСУ цехами осуществляется в соответствии с принципами управления, начиная с подсистемы верхнего (межцехового) уровня, и кончая управляющими элементами, расположенными на самом нижнем (внутрицеховом) уровне иерархии. Иерархическую схему построения КАСУ цехами можно рассматривать, во-первых, как иерархию методов и алгоритмов для решения задач планирования на различных уровнях схемы, во-вторых, как многослойную организацию вычислительного процесса в производственной системе.

Поскольку звенья иерархической схемы являются решающими, то решение задачи нижнего звена зависит от действия высшего звена и, наоборот. Анализ функционирования системы управления производством в иерархической схеме показывает, что при движении от нижних уровней схемы к верхним алгоритмы становятся менее формализованными и требуют более активного участия лица, принимающего решения (ЛПР) и следовательно, более совершенных средств и алгоритмов обмена информацией между человеком и вычислительной системой. Кроме того, при формировании производственной программы высшее звено определяет функции поведения подчиненных звеньев, определяя тем самым долю подчиненных звеньев в успешной деятельности всей системы.

Рассмотрим некоторые формальные методы формирования и корректировки производственной программы, соответствующие принципам управления в иерархической КАСУ цехами.

Используя принципы разделения задач [I], можно указать, что управление в цехе реализуется в два этапа:

- этап формирования производственной программы;
- этап регулирования (управления).

Отсюда следует, что проблема конструирования комплексной системы решается в два этапа, первый из которых - это планирование (построение программной траектории [2]), второй - синтез управления, где каждому уровню в иерархической структуре соответствует набор задач для решения функций планирования, регулирования (управления), согласно предъявленным к данному уровню требованиям для реализации производственной программы.

Планирование рассматривается как предварительное принятие решений о том "что и как" делать в будущей производственной деятельности. При этом определяется производственная программа для участков цеха с указанием объемов и времени выполнения работ.

Существующие в настоящее время на предприятиях неавтоматизированные методы планирования имеют следующие недостатки:

- отсутствие должной взаимоувязки основных показателей, сбалансированность которых нарушается из-за многократных корректировок,

- неоптимальность принятых к исполнению планов в связи с тем, что методы, используемые при их разработке, не позволяют выбрать эффективные направления использования ресурсов и их комбинаций, т.е. не позволяют произвести большой перебор вариантов, принимаемых к исполнению планов.

Внешняя простота в использовании ручных методов планирования приводит к тому, что разработчики АСУ основные усилия направляют на решение задач расчетного характера, не учитывая в полной мере преимущества ЭВМ, с помощью которой решаются проблемы организационного, экономического и математического характера.

Принципиальная трудность планирования заключается в том, что прежде всего необходимо обеспечить взаимоувязку как во времени, так и в пространстве при разработке методов решения задач планирования. При этом производственная про-

грамма цеха считается обоснованной в той мере, в какой она является прогнозом производственно-хозяйственной деятельности цеха.

В традиционных системах автоматического управления (САУ) задача формирования программной траектории формируется следующим образом.

Задача I. В начальный момент времени t_0 система находится в состоянии $X(t_0), X(t_0)=x_1(t_0),\ldots,x_n(t_0)$. Требуется выбрать такое управление $U(t), t_0 \le t \le t_e$, которое переведет объект в заданное конечное состояние $X(t_e)$.

Однако в сложных комплексных системах управления, к классу которых относится КАСУ цехами, в отличие от традиционных САУ конечное состояние не всегда известно. Поэтому наряду с задачей I в таких системах на первом этапе решается задача определения конечного состояния системы, которая в общем случае формулируется следующим образом.

Задача 2. Найти такой оператор $\Lambda : t_e \Rightarrow X(t_e)$, который для рассматриваемого периода функционирования $[t_0, t_e]$ обеспечивает наиболее эффективное интегральное значение на выходе системы.

Процедуру последовательного решения задач 2 и I назовем процессом формирования производственной программы (управляющих воздействий) в КАСУ цехами. Следовательно, этап формирования управляющих воздействий реализуется последовательным решением двух классов задач:

I) задач определения конечного состояния,

2) задач построения оптимальной траектории.

Описанное позволяет сделать следующее определение. Управляющим воздействием называется вводимое в систему предписываемое возмущение, определяющее закон изменения управляемых параметров в рассматриваемый период времени.

Требование к реализации первого и второго этапов управления определяются в основном заложенными в систему принципами управления.

Выделяются два принципа управления комплексными иерархическими системами [3].

I. Принцип жесткого планирования и управления.

2. Принцип гибкого планирования и управления.

В соответствии с этим комплексные иерархические системы условно подразделяются на два типа: [4] комплексная иерархическая система с жестким регламентационным управлением и комплексная иерархическая структура с управлением по целям и стимулам. В первом случае подсистемы высшего (межцехового) уровня, полностью управляют подсистемами более низкого (внутрицехового) уровня, а во втором - за счет ограничений деятельности нижележащих подсистем. Реализация второго принципа связана с тем, что звенья более высокого уровня обусловливают, но не полностью управляют направленной деятельностью звеньев низкого уровня. Решающие звенья (подсистемы) более низкого уровня должны иметь некоторую свободу действия для того, чтобы выбрать свои собственные переменные решения.

Выбор принципа управления предопределяет способ координации в иерархических системах. При этом возможны следующие варианты координации взаимодействия элементов нижележащего уровня [3];

I) координация путем планирования (прогнозирования) взаимодействия;

2) координация путем оценки взаимодействия.

В комплексной АСУ цехами формирование производственной программы основано на предварительном определении всей системы, а также окружающей среды [4], и проводится с учетом следующих характерных особенностей управления в иерархических системах [5].

I) Элементы верхнего (межцехового) уровня связаны с более крупными расчетами и более медленными аспектами управления системой, обмен со средой происходит с меньшей частотой, динамика процесса выражена слабо и велики периоды времени между моментами принятия решений;

2) Задачи управления на верхних уровнях более трудны для количественной формализации ввиду большей неопределенности;

3) Каждая подсистема вышележащего уровня может привлекать в качестве "экспертов" представителей подсистем нижележащих уровней и способна к прогнозированию;

4) Каждая подсистема содержит в качестве образа представление о самой себе и окружающей среде.

Так как задачи принятия решений в комплексных иерархических системах очень сложны, то для их решения требуются специальные упрощения. При исследовании вопросов управления в иерархической системе принимается следующий аксиоматический базис управления иерархическими системами:

Аксиома I. Глобальная задача находится вне системы, так что ни одному из управляющих органов в пределах иерархии не поручается решение глобальной задачи.

Аксиома 2. Каждая подсистема сообщает в вышележащую подсистему свои внутренние возможности.

Аксиома 3. Каждая подсистема информирована о принципах принятия решений всеми остальными подсистемами комплексной подсистемы.

Аксиома 4. Каждая подсистема высшего уровня формирует управляющие воздействия таким образом, чтобы способствовать выполнению собственных целей низших уровней.

Аксиома 5. Достижение глобальной цели осуществляется через действия низших подсистем, которые координируемы относительно глобальной цели.

Аксиома 6. Каждая управляющая подсистема обладает способностью к прогнозу поведения системы и среды.

Аксиома 7. Для каждой подсистемы определен период автономного функционирования.

Аксиома 8. Связь системы с окружающей средой осуществляется через координирующий орган.

В практике планирования используется приближенный метод решения глобальной задачи (например, формирование производственной программы), основанный на декомпозиции цели в иерархию подцелей (дерево подцелей) [6]. Другими словами, решение глобальной задачи происходит в результате решения иерархически расположенных подзадач. Порядок решения подзадач, как правило, соответствует приоритету действий, т.е. подзадача любого уровня содержит в себе ряд неопределенных параметров, конкретизируемых в результате решения подзадач более низкого уровня. Подзадача на нижнем уровне будет пол-

ностью определена только после решения задачи верхнего уровня.

Каждая подсистема S_j^i i-го уровня рассматривается как отображение $S_j^i : V_j^i \times \Phi_{ij} \to V_j^{i-1}$. Это означает, что в каждой подсистеме S_j^i существует семейство задач $P_j^i(S_j^i(V_j^i, \Phi_{ij}))$ таких, что для любого входа V_j^i выход $V_j^{i-1} = S_j^i(V_j^i, \Phi_{ij})$, где Φ_{ij} - информация о состоянии объекта (цеха) или нижележащих подсистем.

Предположим далее, что определена исходная задача:

$$Z_y' = Z\{M(\Omega_y(U)), \mathcal{P}\}_{[t_0, t_e]}$$

при использовании принципа гибкого управления, и

$$Z_y'' = Z\{M(L(U)), \mathcal{P}\}_{[t_0, t_e]}$$

при использовании жесткого управления. Здесь $M(\Omega_y(U))$ - множество допустимых решений (система ограничений) при гибком управлении, $M(L(U))$ - множество решений при жестком управлении, $\mathcal{P}$ - мера эффективности, Ω - целевое множество, L - цель, U - управление.

В зависимости от принципов управления цель системы может быть сформирована или в самой системе (определено конечное состояние) или же задана вышестоящей системой. Таким образом, в первом случае возникает дополнительная задача - задача предписания конечного состояния.

Задача определения (предписания) конечного состояния системы задается тройкой $Z_p = (U, \Omega_0, J)$ и формируется следующим образом: при заданной области задающих воздействий вышестоящей системы U, области возможных состояний Ω_0 на конец интервала автономности $[t_0, t_e]$ определить конечное состояние $X(t_e)$, обеспечивающее наилучшее значение целевой функции J.

В задаче Z_p содержится три основных элемента:

1) область задающих воздействий,
2) область возможных конечных состояний системы,
3) мера эффективности конечных состояний.

Решением этой задачи определяется конечное состояние $X(t_e)$ управляемой системы (желаемый вид выходных величин). Конечное состояние $X(t_e)$ рассматривается в качест-

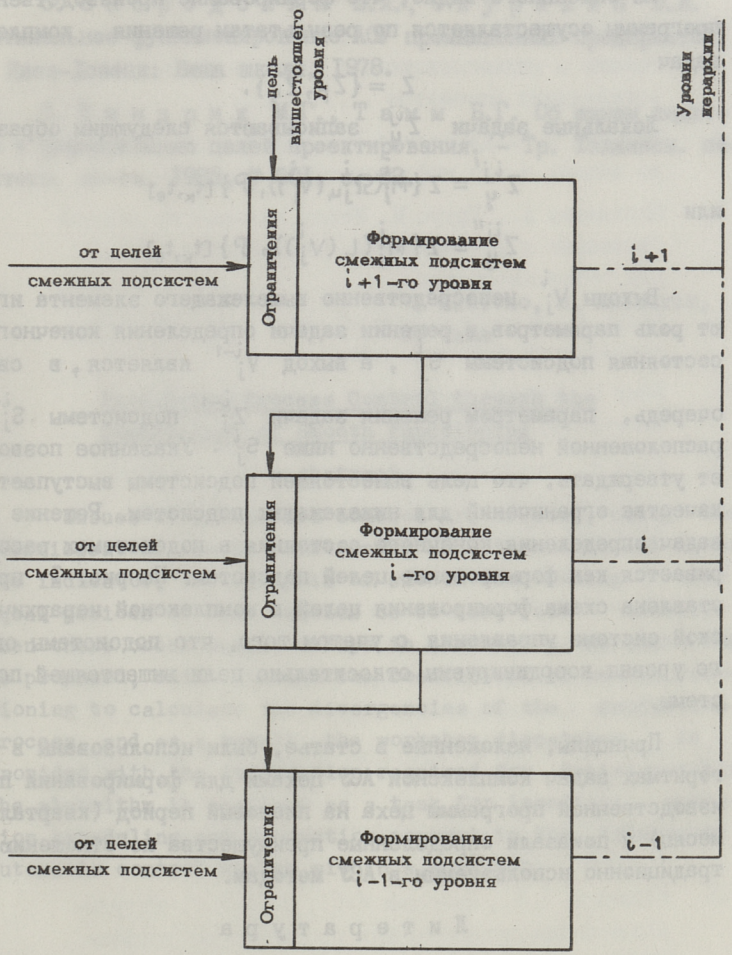


Рис. 1. Схема формирования целей в комплексной иерархической системе управления.

ве цели системы, а достижение этого состояния рассматривается в качестве цели управления.

Из описанного видно, что формирование производственной программы осуществляется по результатам решения комплекса задач

$$Z = \{Z_p, Z_y\}.$$

Локальные задачи Z_y^{ij} записываются следующим образом:

$$Z_y^{ij'} = Z\{M_j^i(\Omega_{ju}^i(V_j^i)), \mathcal{P}\}[t_k, t_e]$$

или

$$Z_y^{ij''} = Z\{M_j^i(L(V_j^i)), \mathcal{P}\}[t_k, t_e].$$

Выходы V_j^i непосредственно вышележащего элемента играют роль параметров в решении задачи определения конечного состояния подсистемы S_j^i, а выход V_j^{i-1} является, в свою очередь, параметром решения задачи Z_j^{i-1} подсистемы S_j^{i-1}, расположенной непосредственно ниже S_j^i. Указанное позволяет утверждать, что цель вышестоящей подсистемы выступает в качестве ограничений для нижележащих подсистем. Решение же задач определения конечного состояния в подсистемах рассматривается как формирование целей подсистем. На рис. I представлена схема формирования целей в комплексной иерархической системе управления с учетом того, что подсистемы одного уровня координируемы относительно цели вышестоящей подсистемы.

Принципы, изложенные в статье, были использованы в алгоритмах задач комплексной АСУ цехами для формирования производственной программы цеха на плановый период (квартал, месяц) и показали определенные преимущества по отношению к традиционно используемым в АСУ методам.

Литература

I. Моисеев Н.Н. Элементы теории оптимальных систем. – М.: Наука, 1975.

2. Иванилов Ю.П., Лотов А.В. Математические модели в экономике. – М.: Наука, 1979.

3. Месарович М., Мако И., Такахара И. Теория иерархических многоуровневых систем. – М.: Мир, 1973.

4. Емельянов С.В. Организационные системы управления: принципы построения структурных схем. – В кн.: Актуальные проблемы управления.–М.: Знание, 1972.

5. Забродский В.А., Скурихин В.И. Оптимальное функционирование АСУ промышленных предприятий. – Киев–Донецк: Вища школа, 1978.

6. Шендрик М.Г., Тамм Б.Г. Об одном подходе к формированию целей проектирования. – Тр. Таллинск. политехн. ин-та, 1985, № 601, с. 83.

V. Mahitko, M. Shendrik,
B. Tamm

Production Process Control through the Coefficient of Smooth Functioning

Abstract

Issues of production control are treated, using the coefficient of smooth functioning as a technical and economic indicator. The analysis of smooth functioning over equal periods of time enables us to take fast measures when there occur output delays in a workshop. An algorithm is proposed, built through the coefficient of smooth functioning to calculate the divergencies of the production process, and as a result, the workshop dispatcher is provided with the information required for decision-making. The algorithm is realized as a task for immediate operation scheduling and production control in the integrated automatic control system within a workshop.

Содержание

руб. 1.30