tanulmányok

99/1980

MTA Számítástechnikai és Automatizálási Kutató Intézet Budapest

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZETE

Д О К Л А Д Ы С И М П О З И У М О В

проводимых 18 - 23 апреля 1977 года в Яхранке и
8 - 13 мая 1978 года в Доновали
во время совещаний НКС СЭВ по теме 1-15.1.

"ТЕОРИЯ АВТОМАТОВ В ПРИМЕНЕНИИ К ПРОЕКТИРОВАНИЮ ДИСКРЕТНЫХ УСТРОЙСТВ И СИСТЕМ"

Tanulmányok 99/1979.

A kiadásért felelős:

DR VÁMOS TIBOR

Szerkesztette:

IVÍCS JÓZSEF

Technikai szerkesztők:

Gabnai Katalin
Kertész Józsefné

ISBN 963 311 096 3
ISSN 0324-2951

Készült a SZÁMOK Reprográfiai Üzemében
80/017

ELŐSZÓ

A *KGST* keretében müködő "Automataelmélet kidolgozása és alkalmazása a diszkrét rendszerek tervezésében". *1-15.1 szakbizottság* 1977. ápr. 18-23. között Lengyelországban a Lengyel Tudományos Akadémia Számitástechnikai Központja, 1978. máj. 8-13. között Csehszlovákiában a Besztercebányai Tanárképző Főiskola szervezésében tartotta ülését. A szakbizottsági ülések a résztvevő országok munkabeszámolóinak és a következő időszakra vonatkozó munkatervének a megvitatása után *szimpóziumként* folytatódtak, ahol a témához kapcsolódó, az egyes országokban elért legujabb eredményekről és a jövőbeni elképzelésekről előadások hangzottak el. Az eddigi gyakorlat szerint az *elhangzott előadások*ról kiadvány készül az MTA SZTAKI gondozásában.
Jelen kiadvány a harmadik összevont (1977-1978).

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ

Глава 2: ДОКЛАДЫ СИМПОЗИУМА НКС 1978 года

СОДЕРЖАНИЕ

Глава 1: ДОКЛАДЫ СИМПОЗИУМА НКС 1977 ГОДА

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1978

ЛООЗЕ, Й.

ДОСТАТОЧНЫЕ УСЛОВИЯ СИНТАКСИЧЕСКОЙ ОДНОЗНАЧНОСТИ СИСТЕМ УРАВНЕНИЙ

Лейпцигский Университет им. Карла Маркса, с. Математики, ГДР

I. ВВЕДЕНИЕ

В рамках теорий формальных языков, языков программирования и построения трансляторов и при рассмотрении конкретных языков программирования, языков для описания структуры дискретных устройств и даже различных исчислений возникает требование уточнения семантики языков, цель которого состоит в том, чтобы разные пользователи языка интерпретировали его одинаково. Это достигается, например, заданием эффективно выполнимого отображения F, которое однозначно соотносит каждому слову языка L некоторый алгоритм фиксированного класса алгоритмов $\mathfrak{A}$. Как правило, такое отображение эффективно задаваемо только тогда, когда наряду с другими требованиями

A. имеется полный обзор всех элементов области определения отображения F, т.е. языка L и

B. доказано, что каждое слово z из L однозначно разложимо в наипростейшие составляющие, семантическое значение которых служит основой для последовательного определения значения слова z.

Пример:
Полный обзор всех выражений исчисления высказываний задается следующим предложением:
H является выражением тогда и только тогда, когда
1. H переменная или
2. существует выражение H' такое что H = ~H' или
3. существуют выражения H' и H" и функтор $\circ$ из множества

$\{\wedge, \vee, \rightarrow, \leftrightarrow\}$ такие что $H = (H' \circ H'')$.

Однозначная разложимость выражений обеспечивается предложением:
Если H является выражением, то имеет место не более одного из выше указанных случаев I., 2. и 3., причем во втором случае выражение H', а в третьем случае выражения H' и H" и функтор ∘ однозначно определены.

Сложность задания обзора всех элементов из L, который показывает, каким образом эти элементы образуются из более простых элементов, значительно зависит от способа определения синтаксиса языка. Оказывается, что надлежашим образом выбранные системы уравнений, уравнения которых имеют вид

"переменная" = "выражение"

выгодны в этом смысле. Встречающиеся в уравнениях выражения имеют в качестве интерпретации множества слов - функторы интерпретируются при этом как операции над множествами слов - и непосредственно отражают структуру слов языка, определенного системой уравнений.

Пример:
Блоки языка "АЛГОЛ 60"[1]) можно определить уравнением

блок = ⟨метка :⟩ начало ⟨описание ;⟩ описание ;
оператор ⟨; оператор⟩ конец

где ⟨ ⟩ обозначает операцию итерации множества слов, которое непосредственно показывает вид блоков.

Простейшие системы уравнений - в виде нормальных форм Бэкуса или контекстно-свободных грамматик - хорошо изучены, но системы уравнений обшего вида, когда интерпретациями выражений являются, как правило, бесконечные множества слов, мало известны, особенно в связи с требованием синтаксической одно-

[1]) "АЛГОЛ 60" - это язык, который определяется нормальной формой Бэкуса из сообшения об АЛГОЛ 60, не учитывая дополнительных контекстных ограничений.

значности. Целью сделанной мною работы, некоторые из основных понятий и результатов которой в нестрогой форме помещены в данной статье, явилось

- точное определение понятия систем уравнений,
- исследование их разрешимости,
- определение понятия синтаксической однозначности систем уравнений, которое отражает требованную выше однозначную разложимость и
- задание ряда эффективных условий синтаксической однозначности, которые позволяют систематический подход к доказательству этого свойства, по крайней мере для т.н. регулярных систем уравнений.

Пусть A некоторый конечный непустой алфавит, состоящий из попарно непересекающихся подалфавитов X терминалов и $Y = \{ y_1, \ldots, y_n \}$ нетерминалов, A^* и X^* множества всех слов конечной длины над этими алфавитами и $\cup$ и $< >$ символы операций объединения и итерации множеств слов. Пусть далее H и L обозначают множества слов над A, $\bar{L}$ n-ки $[L_1, L_2, \ldots, L_n]$ множеств слов, z, u, v слова из A^* и Z слова из X^*. Известно из семиотики, что каждое слово z однозначно представимо в виде

$$z_{0,z} y_{j_1,z} z_{1,z} y_{j_2,z} \ldots y_{j_{s_z},z} z_{s_z,z}$$

при $z_{k,z} \in X^*$ для $0 \leqslant k \leqslant s_z$ и $y_{j_k,z} \in Y$ для $1 \leqslant k \leqslant s_z$.

2. ПОДСТАНОВКА И ОДНОЗНАЧНАЯ ПОДСТАНОВОЧНОСТЬ

Основным средством для исследования систем уравнений является операция подстановки n-ка $\bar{L}$ в множество H, которая обозначается символом S. $S(H,\bar{L})$ – это множество слов над A, которое состоит из всех тех и только тех слов, которые получаются в результате замены в каждом слове из H всех букв $y_1, \ldots, y_n$ произвольными словами из $L_1, \ldots, L_n$ соответственно. При этом возможно заменить одинаковые нетерминалы в разных местах различными словами, т.е. по определению имеет место

$$S(H,\bar{L}) = \bigcup_{z \in H} z_{0,z} L_{j_1,z} z_{1,z} L_{j_2,z} \ldots L_{j_{s_z},z} z_{s_z,z}$$

Для каждого слова z' из $S(H,\bar{L})$ существуют слово z из H и s_z слов z_k из $L_{jk,z}$ такие, что z' получается заменой k-го нетерминала - слева - в z словом z_k для $1 \leq k \leq s_z$. В случае, когда $z, z_1, \ldots, z_{s_z}$ однозначно определены для каждого слова z' из $S(H,\bar{L})$, мы будем сказать, что $\bar{L}$ однозначно подставим в H, кратко $\underline{\text{ESub}}\ H,\bar{L}$. Тогда всякое слово z' из $S(H,\bar{L})$ получается в результате описанной подстановки в точности один раз, или же, другими словами, допускает однозначное разложение в составляющие в соответствии с H и $\bar{L}$.

Для доказательства однозначной подстановочности п-ка $\bar{L}$ в некоторое множество слов H могут быть полезны следующие предложения:

Предложение I

Если H множество слов над алфавитом X или $H = \{y\}$ при $y \in Y$, то всякий п-ок $\bar{L}$ однозначно подставим в множество H, т.е.

$\forall H \bar{L}\ (H \subseteq X^* \dashrightarrow \underline{\text{ESub}}\ H,\bar{L})$

$\forall y \bar{L}\ (y \in Y \dashrightarrow \underline{\text{ESub}}\ \{y\},\bar{L})$

Предложение 2

Если п-ок $\bar{L}$ однозначно подставим в множества H_1 и H_2 и множества $S(H_1,\bar{L})$ и $S(H_2,\bar{L})$ не пересекаются, то $\bar{L}$ однозначно подставим в множество $H_1 \cup H_2$, т.е.

$\forall H_1 H_2 \bar{L}\ (\underline{\text{ESub}}\ H_1,\bar{L} \wedge \underline{\text{ESub}}\ H_2,\bar{L} \wedge S(H_1,\bar{L}) \cap S(H_2,\bar{L}) = \emptyset \dashrightarrow \underline{\text{ESub}}\ H_1 \cup H_2,\bar{L})$

Предложение 3

Если п-ок $\bar{L}$ однозначно подставим в множества H_1 и H_2 и всякое слово z множества $S(H_1,\bar{L})S(H_2,\bar{L})$ единственным образом образуется в результате конкатенации некоторого слова z_1 из $S(H_1,\bar{L})$ и слова z_2 из $S(H_2,\bar{L})$, то $\bar{L}$ однозначно подставим в множество $H_1 H_2$, т.е. при [1]

---

[1] Для свойств "однозначной конкатенарности" и "однозначной итерируемости" имеется ряд достаточных условий.

$\underline{EVk}\ H',H'' \longleftrightarrow \forall z_1 z_2 u_1 u_2 (z_1,u_1 \in H' \wedge z_2,u_2 \in H'' \wedge z_1 z_2 = u_1 u_2 \longrightarrow z_1 = u_1)$

имеет место

$\forall H_1 H_2 \bar{L} (\underline{ESub}\ H_1,\bar{L} \wedge \underline{ESub}\ H_2,\bar{L} \wedge \underline{EVk}\ S(H_1,\bar{L}),S(H_2,\bar{L}) \longrightarrow \underline{ESub}\ H_1 H_2,\bar{L})$

Предложение 4

Если n–ок $\bar{L}$ однозначно подставим в множество H и при $S(H,\bar{L}) \neq \{\varepsilon\}$ итерация $<S(H,\bar{L})>$ является свободной полугруппой с образующими $S(H,\bar{L})$, то $\bar{L}$ однозначно подставим в множество H, т.е. при $\underline{EIt}\ H' \longleftrightarrow \underline{ESub}\ H',<H'>$ имеет место

$\forall H \bar{L} (\underline{ESub}\ H,\bar{L} \wedge \underline{EIt}\ S(H,\bar{L}) \wedge S(H,\bar{L}) \neq \{\varepsilon\} \longrightarrow \underline{ESub}\ <H>,\bar{L})$

3. СИСТЕМЫ УРАВНЕНИЙ

Пусть алфавит B является некоторым расширением алфавита A, содержашим дополнительно символы для операций над множествами слов, напр. < >, ∪, и технические символы, напр. (,). Пусть далее Выр некоторое множество слов над B, элементы которого мы будем называть выражениями и обозначать символом T. Множеством Выр может быть, например, множество регулярных выражений над алфавитом A. Для выражений задана интерпретация I, соотносящая каждому выражению T из Выр однозначно определенное множество слов I(T) над алфавитом A, т.е.

$I : \underline{Выр} \longrightarrow 2^{A^*}$.

Всякий n–ок выражений $\bar{T} = [T_1,\ldots,T_n]$ будем называть системой уравнений, которую можно написать и в более привычной форме

$$\begin{cases} y_1 = T_1 \\ \text{"} \\ y_n = T_n \end{cases}$$

если $= \notin B$ предполагается.

Некоторый n–ок $\bar{L} = [L_1,\ldots,L_n]$ множеств слов называется решением системы уравнений $\bar{T}$, если имеет место

I. $L_j \subseteq X^*$ и

2. $L_j = S(I(T_j), \bar{L})$ для всех j при $1 \leqslant j \leqslant n$.

Для каждой системы уравнений $\bar{T}$ можно, согласно следующему определению, построить некоторый вполне определенный n-ок $\bar{L}(\bar{T})$, который в соответствии с предложением 5 является решением этой системы, причем минимальным.

$$\bar{L}^0(\bar{T}) = [\emptyset, \emptyset, \ldots, \emptyset]$$
$$\bar{L}^{i+1}(\bar{T}) = [S(I(T_1), \bar{L}^i(\bar{T})), \ldots, S(I(T_n), \bar{L}^i(\bar{T}))]$$
$$\bar{L}(\bar{T}) = \bigcup_{i=0}^{\infty} \bar{L}^i(\bar{T})$$

Имеет место

Предложение 5

Для всякой системы уравнений $\bar{T}$ n-ок $\bar{L}(\bar{T})$ является минимальным решением, т.е.

$\bar{L}(\bar{T})$ - решение $\bar{T}$

$\forall \bar{L}'$ ($\bar{L}'$ - решение $\bar{T}$ $\rightarrow$ $\bar{L}(\bar{T}) \subseteq \bar{L}'$)

Далее известны достаточные условия однозначной разрешимости систем уравнений - см. нарп. /3/ - и возможно определить ряд понятий, которые соответствуют некоторым известным для контекстно-свободных грамматик - см. нарп. /1/ - понятиям, как зависимость нетерминалов, эквивалентность и т.п., и для систем уравнений и показать аналогичные предложения.

4. СИНТАКСИЧЕСКАЯ ОДНОЗНАЧНОСТЬ СИСТЕМ УРАВНЕНИЙ

По выше указанному способу построения минимального решения $\bar{L}(\bar{T})$ системы уравнений $\bar{T}$ каждое слово компонентов решения получается в результате последовательной замены нетерминалов в словах множеств $I(T_j)$ словами, которые получились раньше как элементы решения. При этом возможно, что некоторые слова решения получаются по различным путям, или же, что то же самое, могут быть разложены - в соответствии с синтаксисом, заданным системой уравнений - по-разному в составляющие, что противоречит требованию В., сформулированному в введении. Поэтому нам интересны прежде всего те системы уравнений, у которых все слова компонентов решения образуются указанным

способом точно один раз - возможно конечно на разных ступенях i. Такие системы уравнений будем называть синтаксически однозначными. Точное определение этого понятия можно дать подобно понятию однозначности контекстно-свободных грамматик.

Имеет место следующее основное предложение, которое сводит свойство синтаксической однозначности систем уравнений к хорошо изученному свойству однозначной подстановочности.

Предложение 6

Система уравнений $\bar{T} = \lceil T_1, \ldots, T_n \rceil$ синтаксически однозначна тогда и только тогда, когда минимальное решение $\bar{L}(\bar{T})$ этой системы однозначно подставимо в множества $I(T_j)$ для $j = 1, 2, \ldots, n$, т.е. т. и т. т., когда имеет место

$$\forall j \; (\; 1 \leqslant j \leqslant n \; \dashrightarrow \; \underline{\text{ESub}} \; I(T_j), \bar{L}(\bar{T}) \;)$$

5. ПРИМЕНЕНИЕ

Исследование систем уравнений на синтаксическую однозначность с помощью применения предложений 6 и I - 4 эффективно прежде всего для регулярных систем уравнений, т.е. систем $\bar{T}$, у которых все выражения T_j регулярны. В этом случае интерпретации выражений суть регулярные события, которые получаются из элементарных событий $\{a\}$, $a \in A$ - в которые в соответствии с предложением I однозначно подставимы все n -ки $\bar{L}$ - в результате применения конечного числа операций объединения, конкатенаций и итераций, что позволяет применение предложений 2 - 4 к проверке однозначной подстановочности.

Самым сложным рассмотренным примером был язык "АЛГОЛ 60", который определяется как минимальное решение следующей регулярной системы уравнений. Для простоты здесь были исключены из программ код-процедуры и строки.

Система уравнений, определяющая язык "АЛГОЛ 60"

prog = block ∪ compst

block = ⟨label :⟩ unlblock

compst = ⟨label :⟩ unlcomp

basst = ⟨label :⟩ unlbas

condst = ⟨label :⟩ unlcond

forst = ⟨label :⟩ unlfor

unlblock = begin decl ⟨; decl ⟩; ⟨stat ;⟩ stat end

unlcomp = begin ⟨stat ;⟩ stat end

stat = basst ∪ compst ∪ block ∪ condst ∪ forst

unlcond = if Boexp then (block ∪ compst ∪ basst) [else stat] ∪
if Boexp then forst

unlfor = for var := forliel ⟨, forliel ⟩ do stat

forliel = arexp [step arexp until arexp ∪ while Boexp]

unlbas = assst ∪ gotost ∪ dumst ∪ procst

assst = var := ⟨var := ⟩ (arexp ∪ Boexp)

gotost = goto desexp

dumst = ε

procst = iden [(actpar ⟨ (, ∪) let ⟨ let ⟩ : ( ) actpar ⟩)]

actpar = arexp ∪ Boexp ∪ desexp ∪ iden

decl = tydecl ∪ arrdecl ∪ swidecl ∪ prodecl

tydecl = [own] (real ∪ integer ∪ Boolean) ⟨ iden , ⟩ iden

arrdecl = [[own] (real ∪ integer ∪ Boolean)] array arrseg ⟨, arrseg ⟩

arrseg = ⟨ iden , ⟩ iden [arexp : arexp ⟨, arexp : arexp ⟩]

swidecl = switch iden := desexp ⟨, desexp ⟩

prodecl = [real ∪ integer ∪ Boolean] procedure
iden [(iden ⟨ (, ∪) let ⟨ let ⟩ : ( ) iden ⟩)] ;
[value iden ⟨, iden ⟩ ;] ⟨ spec iden ⟨, iden ⟩ ; ⟩ stat

spec = string ∪ switch ∪ label ∪ array ∪ procedure ∪
(real ∪ integer ∪ Boolean) [array ∪ procedure]

arexp = ⟨ if Boexp then siarex else ⟩ siarex

siarex = [+ ∪ −] prim ⟨ ↑ prim ⟩ (x ∪ / ∪ ÷) prim ⟨ ↑ prim ⟩ ⟩ ⟨ (+ ∪ −) . . ⟩

prim = unsnum ∪ var ∪ procst ∪ (arexp)

Boexp = ⟨ if Boexp then siBoex else ⟩ siBoex

siBoex = Bosec ⟨ ∧ Bosec ⟩ ⟨ ∨ Bosec ⟨ ∧ Bosec ⟩ ⟩
⟨ ⊃ Bosec ⟨ ∧ Bosec ⟩ ⟨ ∨ Bosec ⟨ ∧ Bosec ⟩ ⟩ ⟩ ⟨ = ... ⟩

Bosec = [¬] (var ∪ procst ∪ (Boexp) ∪ true ∪ false ∪
siarex (⟨ ∪ ⩽ ∪ = ∪ ⩾ ∪ ⟩ ∪ ≠) siarex)

desexp = ⟨ if Boexp then sideex else ⟩ sideex

sideex = label ∪ (desexp) ∪ iden [arexp]

var = iden [[arexp ⟨ , arexp ⟩]]

label = iden ∪ unsint

iden = let ⟨ let ∪ dig ⟩

unsnum = $_{10}$ [+ ∪ −] unsint ∪ (. unsint ∪ unsint [. unsint]) [$_{10}$ [+ ∪ −] unsint]

unsint = dig ⟨ dig ⟩

let = a ∪ b ∪ c ∪ ... ∪ z ∪ A ∪ B ∪ C ∪ ... ∪ Z

dig = 0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9

В этой записи ∪ является функтором объединения,
< > является функтором итерации,
( ) являются техническими символами, т.е. скобками, а [T] сокрашенно обозначает (ε ∪ T), где ε является символом пустого слова. С целью упрошения чтения вместо нетерминалов употребляются сокрашенные английские названия компонентов решения системы уравнений - см. / 7 / -, которые следует понимать в соответствии с следующей таблицей.

progr	программа
block	блок
compst	составной оператор
basst	основной оператор
condst	условный оператор
forst	оператор цикла
unlblock	непомеченный блок
unlcomp	непомеченный составной
stat	оператор
unlcond	непомеченный условный оператор
unlfor	непомеченный оператор цикла
forliel	элемент списка цикла

assst	оператор присваивания
gotost	оператор перехода
unlbas	непомеченный основной оператор
dumst	пустой оператор
procst	оператор процедуры
actpar	фактический параметр
decl	описание
tydecl	описание типа
arrdecl	описание массивов
arrseg	сегмент массива
swidecl	описание переключателя
prodecl	описание процедуры
spec	спецификация
arexp	арифметическое выражение
siarex	простое арифметическое выражение
prim	первичное выражение
Boexp	логическое выражение
siBoex	простое булевское выражение
Bosec	вторичное логическое выражение
desexp	именующее выражение
sideex	простое именующее выражение
var	переменная
label	метка
iden	идентификатор
unsnum	число без знака
unsint	целое без знака
let	буква
dig	цифра

Доказано, что компоненты решения этой системы совпадают - с исключением строк и код-процедур - с интерпретациями соответствующих нетерминалов нормальной формы Бэкуса из /7/. но эта система синтаксически неоднозначна. После некоторых незначительных изменений всего трех уравнений, вследствие которых "арифметические переменные", "логические переменные", "арифметические процедуры", "логические процедуры", именующие выражения и "идентификаторы процедур" стали синтаксически различимыми, получилась синтаксически однозначная система уравнений. Доказательство синтаксической однозначности проводилось с помошью выше перечисленных предложений и некоторых лемм о скобочной структуре языка.

6. ЛИТЕРАТУРА

/I/ Гинзбург С. Математическая теория контекстно-свободных языков, изд. "Мир", Москва I970

/2/ Maurer H. Theoretische Grundlagen der Programmiersprachen. Theorie der Syntax, BI Hochschultaschenbücher, Band 404, BI-Wissenschaftsverlag, Mannheim 1969

/3/ Редько В.Н. Некоторые вопросы теории языков, журн. "Кибернетика", № 4, Киев I965

/4/ Rohleder H. Über einige Probleme bei der mathematisch exakten Definition der Semantik einer Programmiersprache, EIK 11, Berlin 1975

/5/ Rohleder H. Mathematische Semiotik, Programm und Vortragsauszüge zur Haupttagung der Mathematischen Gesellschaft der DDR 1976, Heft 3

/6/ Starle P.H. Abstrakte Automaten, VEB DVW, Berlin 1969

/7/ Наур П. Алгоритмический язык АЛГОЛ 60. Пересмотренное сообшение, изд. "Мир", Москва I965

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1977

ON FINDING ROOTS ON FINITE-VALUED FUNCTIONS

PAWEŁ KERNTOPF
Computational Centre
Polish Academy of Sciences
00-901 Warsaw, P.O.Box 22
Poland

ABSTRACT

Square-rooting of Boolean functions with respect to substituting a function in place of one of its variables was solved in [1,2,7] by solving Boolean equations or by means of special Boolean operators. These methods are not applicable directly to the general problem in case of arbitrary finite-valued functions. In this paper it is shown that the graph model of composition of functions introduced in [3] provides a solution for the problem of finding roots of arbitrary finite-valued functions with respect to any pattern of iterative substitution of the funciton in place of some of its variables.

1. INTRODUCTION

Consider a function $f(x_1,\ldots,x_n)$ over a finite domain, i.e. $f:A^n \to A$ and the cardinality of A is finite. We shall call f a *finite-valued function* of n variables $x_1,\ldots,x_n$. Let us assume $A=\{0,1,\ldots,k-1\}$, $k \geq 2$. When $k=2$, f will be called a *Boolean function*. If f is a Boolean function then $\bar{f}$ denotes the function $1-f$. The *weight* of a Boolean function f is the number of n-tuples $(a_1,\ldots,a_n)$, $a_j \in A$, $1 \leq j \leq n$, for which $f(a_1,\ldots,a_n) = 1$.

Let x^0,x^1 be a partition of the set $\{x_1,\ldots,x_n\}$ with m variables in x^0, and assume, without loss of generality, that $x^0 = \{x_1,\ldots,x_m\}$. Given functions $g_1,\ldots,g_m$, $g_j:A^n \to A$, $1 \leq j \leq m$, *iterative compositions* of $f:A^n \to A$ with respect to m variables in x^0 and with respect to the sequence of functions $g = (g_1,\ldots,g_m)$ are defined as follows:

$$C^1_{f,g}(x,\ldots,x_n) = f(x_1,\ldots,x_n)$$

$$C^{i+1}_{f,g}(x_1,\ldots,x_n) = C^i_{f,g}(g_1(x_1,\ldots,x_n),\ldots,g_m(x_1,\ldots,x_n),\ x_{m+1}\ ,\ldots,\ x_n)$$

for $i \geq 1$.

Iterative compositions $C^i_{f,g}$ with all $g_1,\ldots,g_m$ equal to f will be called *iterations* of f and denoted by f^i.

By an *i-th root of a function* h we mean a function f such that $h=f^i$ holds identically. Iterations of finite-valued functions correspond to iterative combinational logic circuits, i.e. circuits built up from identical logic gates connected in a regular manner (see [2,6]). Thus, the problem of finding roots of finite-valued functions is interesting from the practical point of view.

It was shown (see [1]) that if $x^0=\{x_1\}$, then for every Boolean function f

$$C^3_{f,(f)} = C^1_{f,(f)}.$$

Note that in this equation (f) denotes a sequence containing only one function, viz. f.

Later, Thayse [2] proved that

$$C^4_{f,g} = C^2_{f,g}$$

for every Boolean function f and every m, $g_j=1$ or $\bar{f}$ for $1 \leq j \leq m$. Thus, for Boolean functions it is sufficient to consider square and cubic roots.

The square-rooting of Boolean functions for $m=1$ was solved in [1,2,7]. It was shown that a Boolean function has a square root with respect to $x^0 = \{x_j\}$ if and only if it is isotone with respect to x_j. When this is the case a parametric solutions given by Rudeanu [7] describing all square roots of a function f is

$$f_0 + tx_j + \bar{f}_0 f_1 \bar{t}\bar{x}_j$$

where $t \leq \bar{f}_0 f_1$ ($g \leq h$ means that if g is equal to 1 then h is equal to 1, too),

$$f_0 = f(x_1,\ldots,x_{j-1},0,x_{j+1},\ldots,x_n),$$

$$f_1 = f(x_1,\ldots,x_{j-1},1,x_{j+1},\ldots,x_n).$$

This implies that f has exactly 2^p square roots, where p is the weight of $\bar{f}_0 f_1$. Rudeanu [7] extended the above solution to functions over an arbitrary Boolean algebra.

In this paper a graph model of functional compoisiton called a composition graph will be applied to finding roots of finite-valued functions. Previous results on finding square roots of Boolean functions were obtained by transformation of Boolean expressions and solving Boolean equations [1,7] or by means of special Boolean operators [2]. These methods cannot be directly applied to the general case of arbitrary finite-valued functions. A composition graph approach provides a simple solution also in the general case.

First, we solve the problem of finding square and cubic roots of Boolean functions in case of substituting a function in place of any number of its variables. Modification of the notion of iterations by allowing substitution of the complement of a function is also discussed. Generalization of our first result to this case isstraight forward. Then, a method of finding solutions in the general case of arbitrary finite-valued functions will be presented.

Some notions and results from [3,4,6] will be briefly described in the next section for convenience of the reader. However, a familiarity with basic concepts of graph theory and Boolean functions theory is assumed.

2. COMPOSITION GRAPHS

A *composition graph* is a quadruple (V,E,L,γ) where V and E are respectively, a finite set of vertices and a set of edges $(E \subseteq V \times V)$ of a directed graph (V,E), every vertex of which has in-degree 1 (i.e. it is the terminal vertex of exactly one edge); L is some set of labels; $\gamma: V \to L$ assigns to each vertex v a label $\gamma(v)$.

Let us define a composition graph $G_{f,g}=(V,E,L,\gamma)$ as follows:

(i) $V = A^n$

(ii) $\langle(a_1,\ldots,a_n),(b_1,\ldots,b_n)\rangle \in E$ if and only if

$$a_j = \begin{cases} g_j(b_1,\ldots,b_n) & \text{for} \quad 1 \leq j \leq m \\ b_j & \text{for} \quad m+1 \leq j \leq n \end{cases}$$

(iii) The label $\gamma(v)$ assigned to $v=(a_1,\ldots,a_n) \in V$ is $f(a_1,\ldots,a_n)$

The *iterations* of a composition graph $G_{f,g}$ are defined in the following way:

$$G^1_{f,g} = G_{f,g}$$

$$G^{i+1}_{f,g} = (V,E,L,\gamma^{i+1}) \qquad \text{for} \quad i \geq 1,$$

where $\gamma^1 = \gamma$, $\gamma^{i+1}(v) = \gamma^i(v')$, and v' is the unique predecessor of v.

For every f and g it follows from the above definition that

$$\gamma^{i+1}((a_1,\ldots,a_n)) = C^{i+1}_{f,g}\,(a_1,\ldots,a_n)$$

and that the composition graph $G^{i+1}_{f,g}$ is obtained from the composition graph $G^{i}_{f,g}$ by moving all labels at a distance 1 along all edges in the direction of arrows (thus, the labels assigned to all vertices with out-degree equal to zero vanish).

Example. Let the function $f:\{0,1,2\}^2 \to \{0,1,2\}$ be defined as in Table 1 and let $m=1$.

Table 1

x_1	x_2	f	f^2
0	0	1	2
1	0	2	1
2	0	1	2
0	1	2	0
1	1	1	1
2	1	0	2
0	2	1	2
1	2	2	0
2	2	0	1

Then $f^2(0,0) = f(f(0,0),0) = f(1,0)$

$f^2(1,0) = f(f(1,0),0) = f(2,0)$

$f^2(2,0) = f(f(2,0),0) = f(1,0)$ etc.

which is symbolized by arrows in Table 1.

Hence, the set of edges of the composition graph $G^{i}_{f},(f),i \geq 1$, will include

$<(1,0)\ ,\ (0,0)>$

$<(2,0)\ ,\ (1,0)>$

$<(1,0)\ ,\ (2,0)>$ etc.

The composition graphs $G^1_{f,(f)}$ and $G^2_{f,(f)}$ are shown in Figure 1 and Figure 2, respectively. Note that the graphs $G^i_{f,(f)}$ are divided into three disjoint parts, each containing all vertices with the same second coordinate. On the basis of Figure 1 it is easy to establish that $f^7 = f$.

Thus, iterations of a finite-valued function can be conveniently generated using the notion of a composition graph. Such a graph is constructed by associating a labeled vertex with each point of the domain and by attaching a directed edge from a vertex v to a vertex $w = (b_1,\ldots,b_n)$ if $v = (f(w),\ldots,f(w),b_{m+1},\ldots,b_n)$. The composition graph associated with f^{i+1} is obtained from it by moving the label assigned to each vertex at the distance i along all paths of length i directed away from the vertex.

Hence, the problem of finding the i-th roots may be reformulated as follows. Given a composition graph $G^i_{f,g}$, $g=(g_1,\ldots,g_m)$, $g_j=f$, $1 \le j \le m$, in which all edges has been erased, find where the edges were located and then move all the labels along edges at the distance i in the direction opposite to arrows, supplying appropriate labels at vertices which have no paths of length i directed away from them. In the next sections of the above problem will be presented.

3. ROOTS OF BOOLEAN FUNCTIONS

The last n-m coordinates of the initial and the terminal vertices of every edge in $G_{f,g}$ coincide. Thus, the graph $G_{f,g}$ is partitioned into 2^{n-m} disjoint parts $P_1,\ldots,P_{2^{n-m}}$ with 2^m vertices in each part. Then, for a Boolean function f and $g=(f,f,\ldots,f)$ the first m coordinates of the initial vertex of any edge may be only all 0's or all 1's. For a given part P_j with the last n-m coordinates equal to $b^j_{m+1},\ldots,b^j_n$ let us denote

$(0,\ldots,0,0,b^j_{m+1},\ldots,b^j_n)$ by v^j_0

$(0,\ldots,0,1,b^j_{m+1},\ldots,b^j_n)$ by v^j_1

.....

.....

.....

$(1,\ldots,1,1,b^j_{m+1},\ldots,b^j_n)$ by $v^j_{2^m-1}$.

Let us consider the transformation of each of the 2^{n-m} parts of the composition graph $G_{f,g}$ corresponding to obtaining f^2 and f^3.

Figure 3 shows four possible types of parts and their transformations ($w_1,\ldots,w_{2^m-2}$ is a permutation of the set $\{v_1,\ldots,v_{2^m-2}\}$). Studying Figure 3 it is easy to see that the necessary and sufficient conditions for existance of a square root of a Boolean function f are for each part P_j as follows:

(a) $f(v_0^j)\bar{f}(v^j_{2^m-1}) = 0$

(b) if $f(v_0^j) = f(v^j_{2^m-1}) = a$ then $f(v_r^j) = a$ for every $1\leq r\leq 2^m-2$.

It follows from the above conditions that for $X^0 = \{x_1,x_2\}$ a function have a square root if and only it it is isotone in the variables x_1 and x_2. For $m\geq 2$ there exist functions which are not isotone in all variables and have square roots.

Now, let us denote $f_0 = f(0,\ldots,0,x_{m+1},\ldots,x_n)$ and $f_1 = f(1,\ldots,1,x_{m+1},\ldots,x_n)$.

The above conditions formulated for individual parts of the composition graph may be transformed into the following ones, more useful for examining functions:

(a) $f_0\bar{f}_1 = 0$

(b) if $\bar{f}_0\bar{f}_1 = 1$ then $f=0$,

(c) if $f_0f_1 = 1$ then $f=1$.

It is easy to observe that to find a square root s of a function f we have to transform labels of each part P_j of the composition graph separately according to the following rules:

(i) if $f(v_0^j)=0$ and $f(v^j_{2^m-1}) = 1$

then $s(v_r^j) = f(v_r^j)$ for every $0 \le r \le 2^m-1$

or $s(v_r^j) = \bar{f}(v_r^j)$ for every $0 \le r \le 2^m-1$

(ii) if $f(v_0^j)=f(v^j_{2^m-1}) = a$

then $s(v_0^j) = s(v^j_{2^m-1}) = a$ and $s(v_r^j) = 0$ or 1

for every $1 \le r \le 2^m-2$.

Thus, for arbitrary m all square roots of a Boolean function f are described by the formula below:

$$\bar{f}_0 f_1 (tf + \bar{t}\bar{f}) + (m_1 u_1 + \ldots + m_{\ell_1} u_{\ell_1}) + (m'_1 u'_1 + \ldots + m'_{\ell_2} u'_{\ell_2})$$

where t is a function of the variables $x_{m+1},\ldots,x_n$, $t \le \bar{f}_0 f_1$

$m_1,\ldots,m_{\ell_1}$ are the minterms of $\bar{f}_0\bar{f}_1$

$m'_1,\ldots,m'_{\ell_2}$ are the minterms of $f_0 f_1$

u_j, u'_j are functions of the variables $x_1,\ldots,x_m$,

$u_j \le (x_1 + \ldots + x_m)(\bar{x}_1 + \ldots + \bar{x}_m)$ for $1 \le j \le \ell_1$

$u'_j \ge x_1 x_2 \ldots x_m + \bar{x}_1\bar{x}_2\ldots\bar{x}_m$ for $1 \le j \le \ell_2$

For m=1 after simplification of the above formula we obtain the formula given by Rudeanu.

The necessary and sufficient conditions for existence of the cubic roots of a Boolean function f are the conditions (b) and (c) given previously for the square r roots. The rules for transformation each part P_j of the graph $G_{f,g}$ to find a cubic root of a function f are:

(i) if $f(v_0^j) \oplus f(v^j_{2^n-1}) = 1$ ($\oplus$ denotes sum modulo 2)

then $s(v_r^j) = f(v_r^j)$ for every $0 \leq r \leq 2^n-1$

(ii) the same as (ii) previously.

Hence, the formula describing all cubic roots may be written as follows:

$$(f_0 \quad f_1)f + (m_1u_1+\ldots+m_{\ell_1}u_{\ell_1}) + (m'_1u'_1+\ldots+m'_{\ell_2}u'_{\ell_2})$$

where notation is the same as above.

Thus, the numbers of the square and cubic roots of a Boolean function f are equal, respectively, to $2^{p+r(2^m-2)}$ and $2^{r(2^m-2)}$, where p is the weight of $\bar{f}_0f_1$, r is the weight of $\bar{f}_0\bar{f}_1+f_0f_1$.

Example. The following function was considered in [1,2,7]:

$$f(x_1,x_2,x_3,x_4) = x_1x_2 + \bar{x}_1x_3 + x_2x_4.$$

As f is not isotone in x_1, square roots of f do not exist if x_1 is included in x^0 and $m=2$. Let us take $x^0 = \{x_2,x_4\}$.

Now $f_0=\bar{x}_1x_3$ and $f_1 = 1$,

$f_0\bar{f}_1 = 0$ so the condition (a) is satisfied

$\bar{f}_0\bar{f}_1 = 0$ so the condition (b) is satisfied

$f_0f_1=\bar{x}_1x_3$ and for $x_1=0$, $x_3=1$ the equation $f=1$ holds identically, hence the condition (c) is also satisfied.

Here $p=1$ and $r=3$. Thus, there will be 32 square roots and 4 cubic roots of f. We have $\bar{f}_0 f_1 = x_1+\bar{x}_3 \geq t$, $l_1=0$, $l_2=1$, $m_1'=\bar{x}_1 x_3$, $u_1' \geq \bar{x}_2\bar{x}_4+x_2x_4$. The following formulae describe all square roots s_i and cubic roots c_j of f:

$$s_i = tf + (x_1+\bar{x}_3)\bar{t}\bar{f} + \bar{x}_1 x_3 u_1' \qquad 1 \leq j \leq 32$$

$$c_j = (x_1+\bar{x}_3)f + \bar{x}_1 x_3 u_1' \qquad 1 \leq j \leq 4$$

where

$$u_1' = \bar{x}_2\bar{x}_4 + x_2x_4$$

$$\text{or} \quad \bar{x}_2\bar{x}_4+x_2x_4+\bar{x}_2x_4 = \bar{x}_2+x_4$$

$$\text{or} \quad \bar{x}_2\bar{x}_4+x_2x_4+x_2\bar{x}_4 = x_2+\bar{x}_4$$

$$\text{or} \quad \bar{x}_2\bar{x}_4+x_2x_4+\bar{x}_2x_4+x_2\bar{x}_4 = 1$$

$$t = 0,\ \bar{x}_1\bar{x}_3,\ \bar{x}_1x_3,\ x_1x_3,\ \bar{x}_1\bar{x}_3+x_1x_3,\ x_1,\ \bar{x}_3 \quad \text{or} \quad \bar{x}_1+\bar{x}_3.$$

Let $x^0=\{x_1,x_2,x_3\}$. Then $f_0=0$ and $f_1=1$, so the conditions (a)-(c) are obviously satisfied. Here $f_0f_1=0$, $\bar{f}_0\bar{f}_1=0$, $\bar{f}_0f_1=1$, hence $r=0$, $p=2$, $t=0$, $\bar{x}_4$, x_4 or 1. So we have four square roots: $\bar{f}$, $\bar{x}_4f+x_4\bar{f}$, $x_4f+\bar{x}_4\bar{f}$ and f. The unique cubic root is the function f itself.

Similarly, in case $x^0=\{x_1,x_2,x_3,x_4\}$ we have $f_0=0$ and $f_1=1$ again, $r=0$, $p=1$, $t=0$ or 1, so there are two square roots equal to f and $\bar{f}$. The only cubic root is equal to f.

Now , let us generalize the notion of iterations of a Boolean function f by allowing that $g_j=f$ or $\bar{f}$, $1 \leq j \leq m$, and let $g=(f^{a1}, \ldots, f^{am})$ where $a_j \in \{0,1\}$ for $1 \leq j \leq m$, $f^0=\bar{f}$, $f^1=f$. Then, the first m coordinates of the initial vertex of any edge may be only $\bar{a}_1,\ldots,\bar{a}_m$ or $a_1,\ldots,a_m$. All previous considerations on roots of Boolean functions may be applied to this case if we change:

(1) coding of the vertices in each part P_j of the composition graph $G_{f,g}$ so that

v_0^j will denote $(\overline{a_1},\ldots,\overline{a_{m-1}}, \overline{a_m}, b_{m+1},\ldots,b_n)$

v_1^j will denote $(\overline{a_1},\ldots,\overline{a_{m-1}}, a_m, b_{m+1},\ldots,b_n)$

.....

.....

.....

$v_{2^m-1}^j$ will denote $(a_1,\ldots,a_{m-1},a_m,b_{m+1},\ldots,b_n)$

(2) the meaning of f_0 and f_1 in the following way:

$$f_0 = f(\overline{a_1},\ldots,\overline{a_m},x_{m+1},\ldots,x_n)$$

$$f_1 = f(a_1,\ldots,a_m,x_{m+1},\ldots,x_n)$$

(3) the bounds for u_j and u_j' in the following way:

$$u_j \leq (x_1^{\overline{a_1}}+\ldots+x_m^{\overline{a_m}})(x_1^{a_1}+\ldots+x_m^{a_m}) \qquad 1 \leq j \leq \ell_1$$

$$u_j' \geq x_1^{\overline{a_1}} \ldots x_m^{\overline{a_m}} + x_1^{a_1} \ldots x_m^{a_m} \qquad 1 \leq j \leq \ell_2$$

4. ROOTS OF ARBITRARY FINITE-VALUED FUNCTIONS

It was shown (see [3]) that for every finite-valued function f and every partition x^0,x^1 of its set of variables there exist integers p and h such that

$$f^{p+h} = f^h$$

where $1 \leq h \leq k$, $1 \leq p \leq$ the least common multiple of $1,2,\ldots,k$. Thus, it is sufficient to consider i-th roots for

$$i \leq \text{l.c.m } \{1,2,\ldots,k\} + k - 1.$$

A method of finding roots of arbitrary finite-valued functions may be obtained by the same argument as in the case of Boolean functions. The graph $G_{f,g}$ will be partitioned into k^{n-m} disjoint parts $P_1,\ldots,P_{k^{n-m}}$ with k^m vertices in each part. When $g=(f,f,\ldots,f)$, the first m coordinates of the initial vertex of any edge coincide. Vertices having identical first m coordinates will be called *diagonal*. Only these vertices may belong to any cycle.

Let us sum up some properties which will be used in developing our method:

(P1) If there exist a path from a vertex $v=(a,\ldots,a,b_{m+1},\ldots,b_n)$ to a vertex w in the graph $G^i_{f,g}$, $i \geq 1$, then $\gamma(w)=a$.

(P2) If the set of vertices belonging to a cycle is $V_c=\{v_1,\ldots,v_\ell\}$, $1 \leq \ell \leq m$, where $v_j=(a_j,\ldots,a_j,b_{m+1},\ldots,b_n)$, $1 \leq j \leq l$, then the set of their labels is $\{a_1,\ldots,a_\ell\}$.

(P3) In each cycle of $G^i_{f,g}$ every label is moved at the distance $i \pmod q$ where q is the length of the cycle.

Given the set of all diagonal vertices V_j of the part P_j together with their labels, a subset V'_j of V_j such that

(i) if $v=(a,\ldots,a,b_{m+1},\ldots,b_n) \in V'_j$ then $(\gamma(v),\ldots,\gamma(v),b_{m+1},\ldots,b_n) \in V'_j$,

(ii) there does not exist a subset $V''_j \subset V'_j$ satisfying (i) will be called an *elementary* cycle. A cycle $v_{11},\ldots,v_{1z},v_{21},\ldots,v_{2z},\ldots,v_{y1},\ldots,v_{yz}$ where $v_{j1},\ldots,v_{jz}$, $1 \leq j \leq y$, is an elementary cycle will be called a *complex* cycle. It can be proved that all cycle in graphs $G^i_{f,g}$ are elementary or complex ones.

Given a function f our method of finding i-th roots of f consists of the following steps for each part P_j:

Step 1. Find all elementary and complex cycles having the property P3. Let the set of all their vertices be W.

Step 2. For each partition of the set W into cycles found in the previous step construct paths directed away cycle vertices in such a way that part P_j will have the property P1.

Step 3. Move all the labels in P_j along edges at the distance i in the direction opposite to arrows, supplying appropriate labels at vertices which have no paths of length i directed away from them.

Example. Let the function $f(x_1,x_2):\{0,1,2\}^2 \to \{0,1,2\}$ be defined as in Table 2. Find square roots of f with respect to $\{x_1,x_2\}$.

Table 2

x_1 \ x_2	0	1	2
0	1	0	1
1	0	1	0
2	1	1	1

The graph $G_{f,(f,f)}$ consists of one part. Let us apply our method to it.

Step 1. In this case there exists only one elementary cycle. It is the cycle of length 1 containing the vertex (1,1).

Step 2. As the labels of some vertices are equal to 0, there must be a path of length 2 directed away from the vertex (0,0). The only possibility for this is to insert in the graph the edges: $<(1,1)\ ,\ (0,0)>$, $<(0,0),(2,2)>$, $<(2,2)\ ,\ (0,1)>$, $<(2,2)\ ,\ (1,0)>$, $<(2,2),(1,2)>$. The other vertices, labeled with 1, may be connected to vertices (0,0) or (1,1). Thus, we obtain eight graphs shown together in Figure 4.

Step 3. The result of performing this step is shown in Figure 5. The eight square roots of f can be described by the use of Table 3.

Table 3.

x_1 \ x_2	0	1	2
0	1	2	0/1
1	2	1	2
2	0/1	0/1	0

REFERENCES

[1] Reischer C., Simovici D.A., "Associative algebraic structures in the set of Boolean functions and some applications in automata theory", IEEE Trans. Computers, vol.C-20, March 1971, pp.298-303.

[2] Thayse A., "On some iteration properties of Boolean functions", Philips Res. Repts., vol.28, No.2, April 1973, pp.107-119.

[3] Kerntopf P., "On the periodicity of an iteration of finite-valued functions", Proc. 12th Annual Allerton Conference on Circuit and System Theory, Monticello, Illinois, October 2-4, 1974, pp.115-119.

[4] Kerntopf P., "Iterative compositions of Boolean functions", Proc. 8th Asilomar Conference on Circuits, System and Computers, Pacific Grove, California, December 3-5, 1974, pp.78-81.

[5] Rudeanu S., "Boolean functions and equations", North-Holland Publ. Co,, Amsterdam - London 1974.

[6] Kerntopf P., "Composition graphs and their applications", Proc. Conference on Digital Computer Organization and Microprogramming, Warsaw, Poland, September 24-26, 1975, to appear (in Polish).

[7] Rudeanu S., "Square roots and functional decompositions of Boolean functions", IEEE Trans. Computers, vol.C-25, May 1976, pp.528-532.

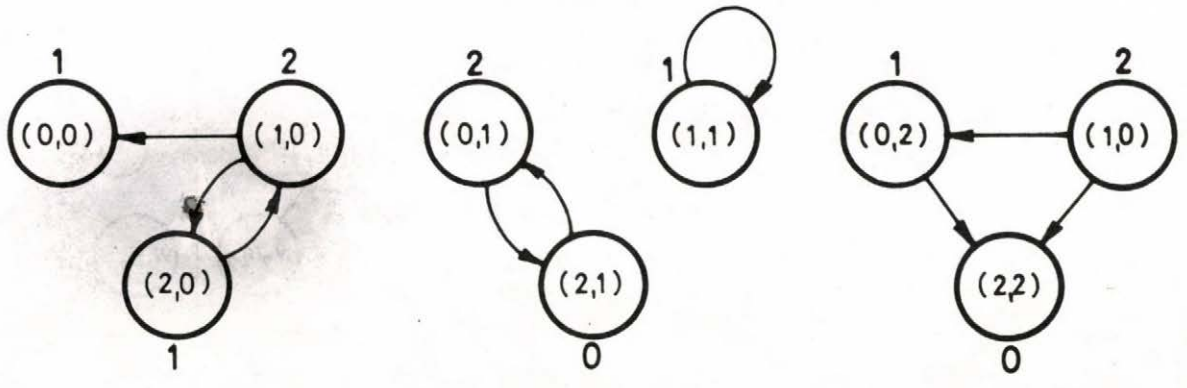

1
2
(0,0)
(1,0)
(2,0)
1
2
1
(0,1)
(1,1)
(2,1)
0
1
2
(0,2)
(1,0)
(2,2)
0

Fig. 1.

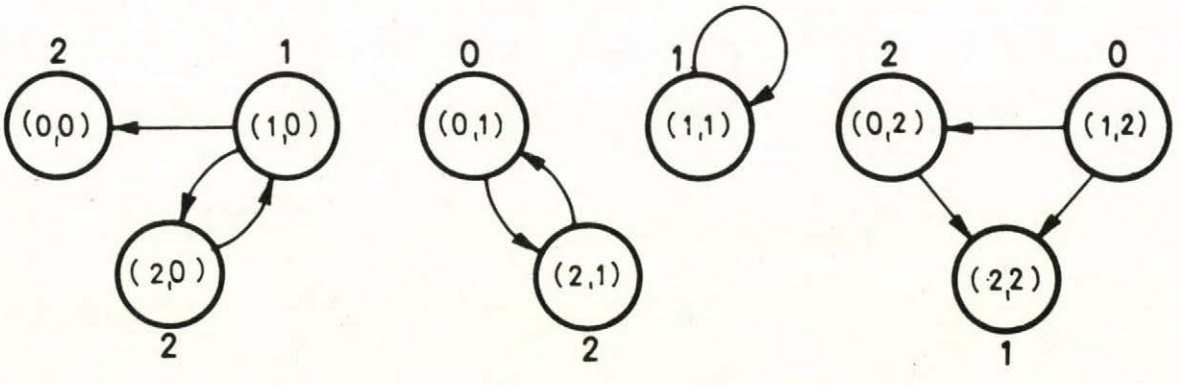

2
1
(0,0)
(1,0)
(2,0)
2
0
1
(0,1)
(1,1)
(2,1)
2
2
0
(0,2)
(1,2)
(2,2)
1

Fig. 2.

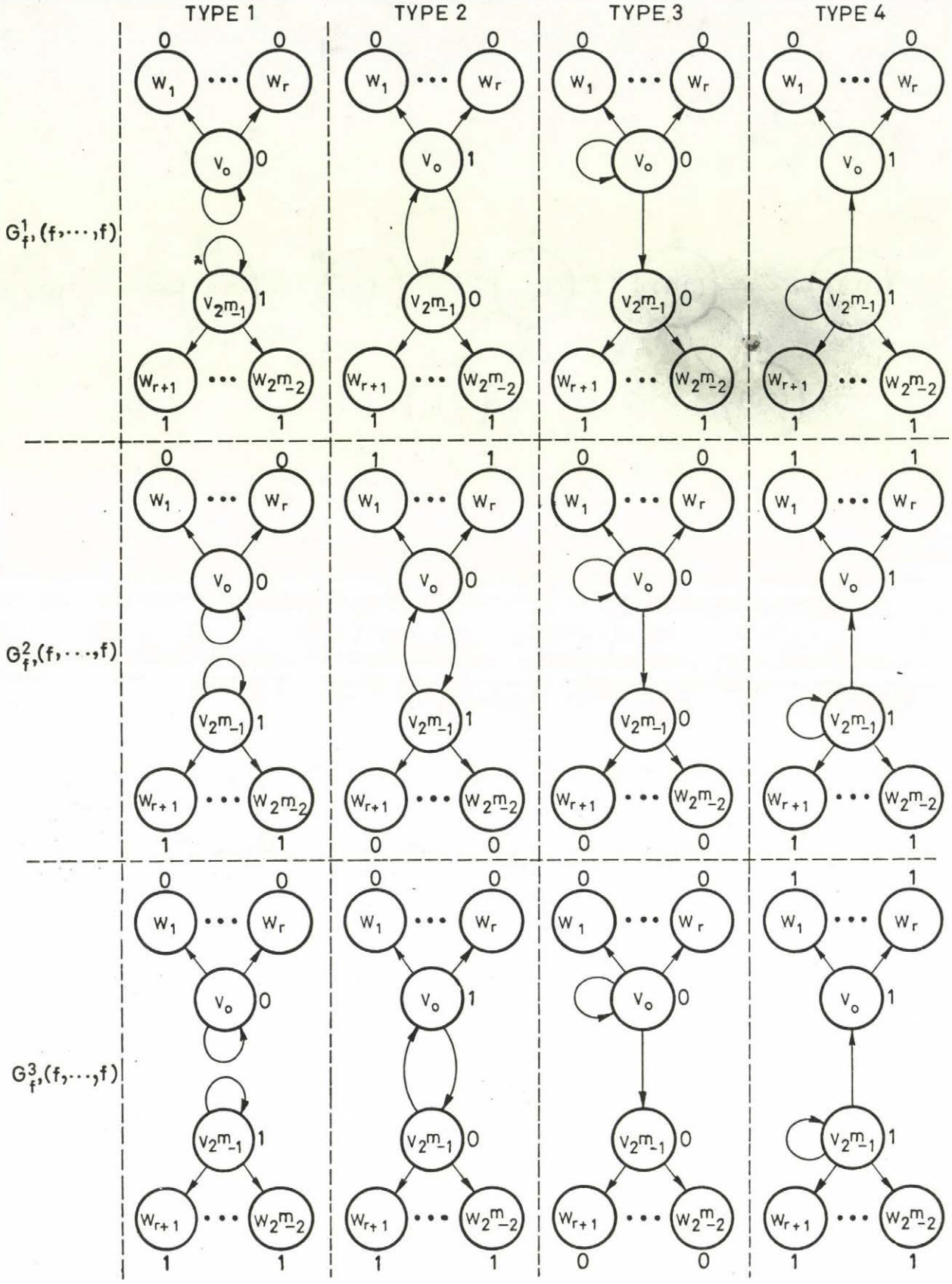

TYPE 1
TYPE 2
TYPE 3
TYPE 4
$G^1_f,(f,\cdots,f)$
$G^2_f,(f,\cdots,f)$
$G^3_f,(f,\cdots,f)$
w_1 w_r v_o v_{2^m-1} w_{r+1} w_{2^m-2}

Fig. 3.

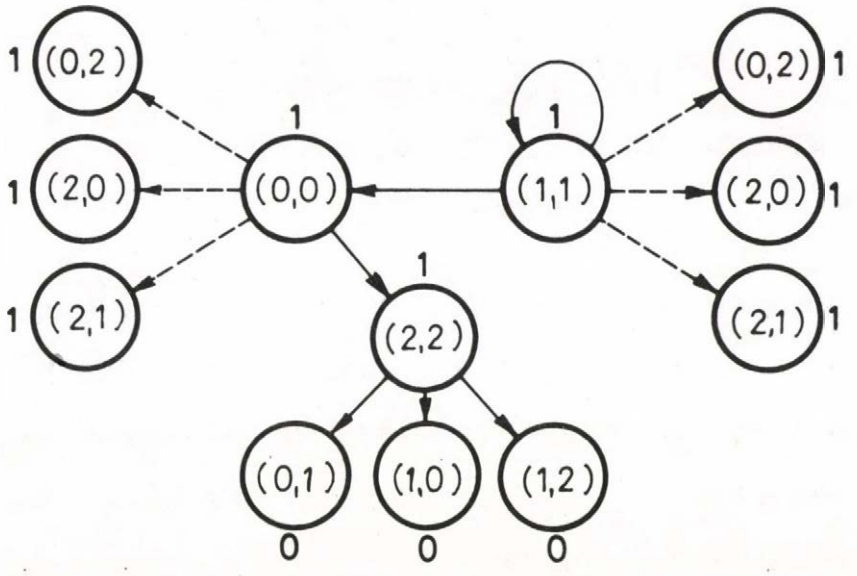

1
(0,2)
1
(2,0)
1
(2,1)
1
(0,0)
1
(1,1)
(0,2)
1
(2,0)
1
(2,1)
1
1
(2,2)
(0,1)
(1,0)
(1,2)
0
0
0

Fig. 4.

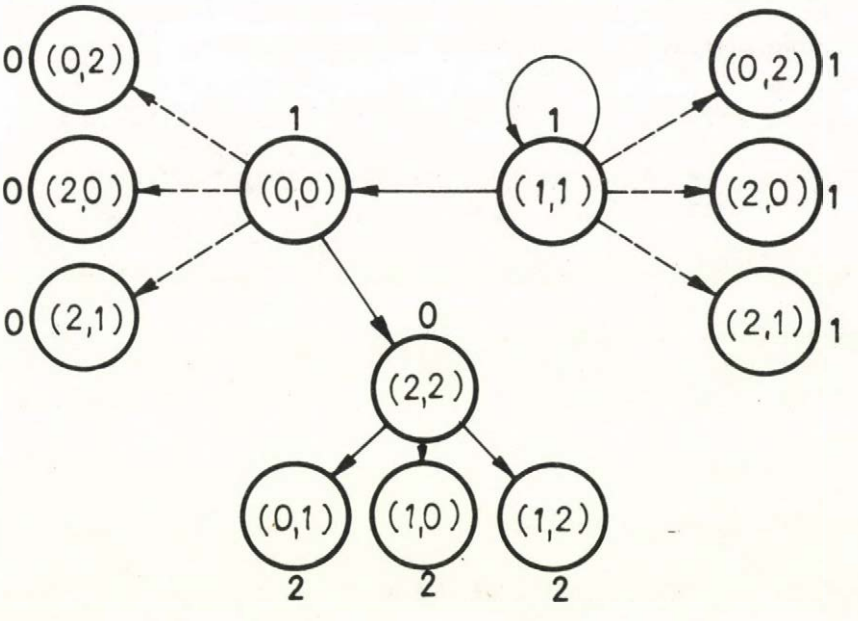

0
(0,2)
0
(2,0)
0
(2,1)
1
(0,0)
1
(1,1)
(0,2)
1
(2,0)
1
(2,1)
1
0
(2,2)
(0,1)
(1,0)
(1,2)
2
2
2

Fig. 5.

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1977

Болеслав Миколайчак

Альгебраическая теория обобщенных разширений конечных автоматов

Институт Автоматики
Познаньский Политехнический Институт
Познань, Польша

Исследования по альгебраической теории конечных автоматов начались в конце 50-тых лет работами Рабина-Скотта, Гартманиса и Веега а по алгебраической теории периодических автоматов работами Гилла и Флексера. В 1969 году были введены Гржимала-Буссем понятия периодической суммы и разширения автомата [1]. С этой времени были получены различные результаты и сделаны различные обобшения [2:7]. Проблемы связанные с периодическими суммами и разширениями очень важные с теоретической и практической точки зрения. Эти понятия представляют математические модели промышленных процесов с переменной структурой или, с другой точки зрения, модели математических машин с переменной структурой, или модель акцептора семьи регульярных языков.

Сначала введём некоторые понятия и обозначения приняемые в этой работе.

Автоматом будем называть упорядоченную тройку $A=(S,\Sigma,M)$, где

S - конечное, непустое множество состояний,

Σ - конечное, непустое множество входов,

M - функция переходов определённа на $S\times\Sigma \xrightarrow{B} S$.

Под точно периодическим автоматом V понимаем упорядоченную тройку (S',Σ,M'), где S' конечная последовательность конеч-

ных, непустых множеств состояний $S'_0, S'_1, \ldots, S'_{T-1}$, Σ конечное непустое множество входов, M' конечная последовательность функций переходов $M'_0, M'_1, \ldots, M'_{T-1}$, где

$$M'_t : S'_t \times \Sigma \longrightarrow S'_{t+1 (\mathrm{mod}\ T)} \quad \text{для } t \in \{0,1,\ldots,T-1\}.$$

Число T называется периодом автомата V . Если $T=1$,то периодический автомат является обыкновенным автоматом.

Путь R будет отношением между элементами свободной полугруппы I определённым следующим видом:

для любого $x,y \in I$ $\quad xRy \iff$ для любого $s \in S$ $M(s,x)=M(s,y)$.

Класс эквивалентности этого отношения, которому принадлежит $x \in I$ будем обозначать $\bar{x}$. Множество всех классов эквивалентности отношения R будет обозначено $\bar{I}$. $\bar{I}$ с операцей сложения класс эквивалентности $\bar{x} \cdot \bar{y} = \overline{xy}$, для каждого $\bar{x}, \bar{y} \in \bar{I}$, создавает полугруппу называемой характеристической полугруппой автомата.

Фиксированным аналогом V^* периодического автомата $V=(S, \Sigma, M')$ называем обыкновенный автомат (S^*, Σ, M^*), где

$$S^* = \bigcup_{t=0}^{T-1} S'_t \quad \text{и} \quad M^* : S^* \times \Sigma \longrightarrow S$$

является функцией переходов определённой для любого $s \in S'_t$ и $t \in \{0,1,\ldots,T-1\}$ следующим видом $M^*(s,\sigma) = M'_t(s,\sigma)$.

Пусть $A_0=(S_0, \Sigma, M_0), A_1=(S_1, \Sigma, M_1), \ldots, A_{r-1}=(S_{r-1}, \Sigma, M_{r-1})$ будут обыкновенными автоматами такими, что $|S_0|=|S_1|=\ldots=|S_{r-1}|$.

Пусть $\varphi_0: S_0 \longrightarrow S_1$, $\varphi_1: S_1 \longrightarrow S_2$, ..., $\varphi_{r-1}: S_{r-1} \longrightarrow S_0$, $h: \Sigma \to \Sigma$ будут взаимооднозначными отображениями. Тогда обобщенной периодической суммой автоматов $A_0, A_1, \ldots, A_{r-1}$ связанной с отображениями $\varphi_0, \varphi_1, \ldots, \varphi_{r-1}$ является периодический автомат $V^+=(S^+, \Sigma, M^+)$ с периодом r , где S^+ является последовательностью $S_0, S_1, \ldots, S_{r-1}$ и M^+ является последовательностью M_0^+ , $M_1^+, \ldots, M_{r-1}^+$, где для любого $s \in S_i$, $\sigma \in \Sigma$, $i=0,1,\ldots,r-1$

имеем

$$M_i^{+}(s,\sigma) = \varphi_i M_i(s,h(\sigma)) .$$

Пусть $A_0=(S_0,\Sigma,M_0)$ обыкновенный автомат и $g_0,g_1,\ldots,g_{r-1}$ будут соответственными изоморфизмами автомата A^0 на автоматы $A_0=(S_0,\Sigma,M_0), A_1=(S_1,\Sigma,M_1),\ldots,A_{r-1}=(S_{r-1},\Sigma,M_{r-1})$. Обобщенным разширением автомата A_0 связанным с изоморфизмами $g_0, g_1,\ldots,g_{r-1}$ и функцей h называем периодический автомат $V=(S',\Sigma,M')$ с периодом $T=r$, где $S'=(S'_0,S'_1,\ldots,S'_{r-1})$, $M'=(M'_0,M'_1,\ldots,M'_{r-1})$ а функция переходов определённа следующим видом

$$M'_i(s,\sigma) = g_{i+1(\mathrm{mod}\ r)}(g_i)^{-1} M_i(s,h(\sigma))$$

для любого $s\in S$, $\sigma\in\Sigma$, $i\in\{0,1,\ldots,r-1\}$.

Для сокрашения будем называть обобщенное разширение автомата A_0 связанное с изоморфизмами $g_0,g_1,\ldots,g_{r-1}$ и функцией h обобщенным разширением и будем обозначать $V=\mathrm{ext}_{r,h}A_0$. Периодический автомат V называется приводимым к обыкновенному автомату A_0 тогда и только тогда, когда является обошенным разширением автомата A_0. Этот факт будем обозначать $A_0=\mathrm{red}_{r,h}V$.

Если в определении обобщенной периодической суммы принимаем

$$\varphi_i = g_{i+1(\mathrm{mod}r)}(g_i)^{-1} \quad \text{для} \quad i=0,1,\ldots,r-1,$$

то такая обобщенная периодическая сумма является обобщенным разширением автомата. Если h=id. , то обобщенная периодическая сумма (обобщенное разширение) называется периодической суммой (разширением).

Теорема 1 Путь $A_0=(S_0,\Sigma,M_0)$ и $A_1=(S_1,\Sigma,M_1)$ будут обыкновенными автоматами такими, что $\overline{I(A_0)} = \overline{I(A_1)}$; тогда подполугруппа характеристической полугруппы $\overline{I(V^*)}$ фиксированного а-

налога $V^* = (S^*, \Sigma, M^*)$ периодической суммы $V = (S^+, \Sigma, M^+)$ автоматов A_0 и A_1 связанна с функциями f_0, f_1 является изоморфична с $\overline{I(A_0)} = \overline{I(A_1)}$.

Доказательство. Для автоматов A_0 и A_1 имеем следующие отношения Мыгилла-Нероде

для A_0 $x \equiv y \iff$ для любого $s \in S_0$ $M_0(s,x) = M_0(s,y)$

для A_1 $x \equiv y \iff$ для любого $s \in S_1$ $M_1(s,x) = M_1(s,y)$

потому, что $\overline{I(A_0)} = \overline{I(A_1)}$. Для фиксированного аналога V^* мы имеем $xRy \iff f_i M_i(s^*,x) = f_i M_i(s^*,y)$ для любого $s^* \in S^* \iff$

$f_0 M_0(s,x) = f_0 M_0(s,y), s \in S_0$ и $f_1 M_1(s';x) = f_1 M_1(s';y)$ $s' \in S_1$

$\iff M_0(s,x) = M_0(s,y)$, $s \in S_0$ и $M_1(s';x) = M_1(s';y)$ $s' \in S_1$

Тем же самым мы показали, что если $\bar{x} \in \overline{I(A_0)} = \overline{I(A_1)}$, то $\bar{x} \in \overline{I(V^*)}$ и $\overline{I(A_0)} = \overline{I(A_1)} \subseteq \overline{I(V^*)}$. Это обозначает, что

$$\overline{I(A_0)} = \overline{I(A_1)} \cong \bar{I} \in \overline{I(V^*)} \quad .$$

Следствие I. Этот результат может быть обобшен на периодическую сумму r автоматов.

Следствие 2. Характеристическая полугруппа $\overline{I(V^*)}$ фиксированного аналога разширения автомата A_0 является в следующей связи с характеристической полугруппой $\overline{I(A_0)}$

$$\overline{I(A_0)} \cong \bar{I} \subseteq \overline{I(V^*)}$$

($\cong$ обозначает изоморфизм полугрупп) .

Пример I. Пусть $A_0 = (S_0, \Sigma, M_0)$, $A_1 = (S_1, \Sigma, M_1)$ такие, что

$S_0 = \{s_1^0, s_2^0, s_3^0\}$, $S_1 = \{s_1^1, s_2^1, s_3^1\}$, $\Sigma = \{0,1\}$

$M_0(s_1^0, 0) = s_1^0$, $M_0(s_2^0, 0) = s_3^0$, $M_0(s_3^0, 0) = s_2^0$

$M_0(s_1^0, 1) = s_2^0$, $M_0(s_2^0, 1) = s_1^0$, $M_0(s_3^0, 1) = s_3^0$

$M_1(s_1^1, 0) = s_1^1$, $M_1(s_2^1, 0) = s_3^1$, $M_1(s_3^1, 1) = s_2^1$

$M_1(s_1^1,1)=s_2^1$, $M_1(s_2^1,1)=s_1^1$, $M_1(s_3^1,1)=s_3^1$

$$\gamma_0=\begin{pmatrix} s_1^0 & s_2^0 & s_3^0 \\ s_1^1 & s_2^1 & s_3^1 \end{pmatrix} \qquad \gamma_1=\begin{pmatrix} s_1^1 & s_2^1 & s_3^1 \\ s_1^0 & s_2^0 & s_3^0 \end{pmatrix}$$

В этом случае $\overline{I(A_0)}=\overline{I(A_1)}=\overline{I(V^*)}=\{0,1,00,01,10,010\}$.

Теорема 2. (о редукции обобщенного расширения к обыкновенному разширению) Пусть $A_0=(S_0,\Sigma,M_0)$ будет обыкновенным автоматом, где $S_0=\{s_1,s_2,\ldots,s_n\}$, $B=(\bar{S}_0,\Sigma,N)$ его h-преобразованием таким, что $\bar{S}_0=\{\bar{s}_1,\bar{s}_2,\ldots,\bar{s}_n\}$. Пусть $ext_{r,h}A=(S',\Sigma,M')$ будет обошенным разширением связанным с изоморфизмами $g_0,g_1,\ldots,g_{r-1}$ и функцией h^{-1} ($ext_{r,h}A)^*=(S'^*,\Sigma,M'^*)$ является фиксированным аналогом) ; пусть $ext_r\ B=(S',\Sigma,N')$ будет обыкновенным разширением связанным с изоморфизмами $\bar{g}_i$, $i=0,1,\ldots,r-1$ ($(ext_r\ B)^*=(\bar{S}'^*,\ ,N'^*)$ является фиксированным аналогом) .

Тогда автоматы $(ext_{r,h}A)^*$ и $(ext_r\ B)^*$ изоморфны .

h -преобразованием автомата $A=(S,\Sigma,M)$ ($h:\Sigma\longrightarrow\Sigma$ является взаимооднозначным преобразованием) называется автомат $B=(S,\Sigma,M_h)$ такой, что

$M_h(s,\sigma)=M(s,h(\sigma))$ для любого $s\in S$, $\sigma\in\Sigma$.

Доказательство. Пусть определим преобразование между множествами состояний $\Theta:S'^*\longrightarrow\bar{S}'^*$ следующим видом $\Theta(s_j^i)=\bar{s}_j^i$ для $i=0,1,\ldots,r-1$ $j=1,2,\ldots,n$. Конечно Θ является взаимнооднозначным преобразованием. Осталось доказать свойство закрепления операции

$$\Theta(M_i(s_j^i,\sigma))=\Theta(g_{i+1(\mathrm{mod}\ r)}M((g_i)^{-1}(s_j^i),h^{-1}(\sigma)))=$$
$$=\Theta(g_{i+1(\mathrm{mod}\ r)}M(s_j,h^{-1}(\sigma)))=\bar{g}_{i+1(\mathrm{mod}\ r)}(N(\bar{s}_j,hh^{-1}(\sigma)))=$$
$$=\bar{g}_{i+1(\mathrm{mod}\ r)}(N(\bar{s}_j,\sigma))=N_i(\bar{g}_i(\bar{s}_j),\sigma)=N_i(\bar{s}_j^i,\sigma)=N_i(\Theta(s_j^i),\sigma).$$

Следствие. $(ext_{r,h}\ A)^*\cong(ext_r\ h^{-1}(A))^*$; где

$(ext_{r,h}\ A)^*$ -фиксированный аналог обобщенного разшипения автомата A,

$(ext_r\ h^{-1}(A))^*$ - фиксированный аналог обыкновенного разширения h^{-1}-преобразования автомата A;

это обозначает,что различные свойства обобщенных разширений обобщенных периодических сумм возможно исследовать с помошью обыкновенного разширения периодической суммы и h^{-1}-преобразования.

Теперь некоторые альгебраические свойства разширений конечных автоматов будут использованы для исследований линейной и разширенной линейной реализации автоматов. Начнём с добавительных определений и обозначений.

Автомат $A=(S,\Sigma,M)$ называется линейно(разширенно линейно) реализованным над GF(p) тогда и только тогда, когда существуют две иниекции α_1 и α_2 ,которые отображают соответственно множества S и Σ в множества последовательности длины n и k элементов из GF(p) и если существуют две(три) матрицы над GF(p) A, B(φ_0) такие, что

$$\alpha_1(M(s,\sigma))= A\alpha_1(s) + B\alpha_2(\sigma) + \varphi_0$$

для любого $s\in S$, $\sigma\in\Sigma$, где + обозначает аддитивную операцию в GF(p) .

Автомат $A=(S,\Sigma,M)$ называется пермутационным тогда и только тогда, когда для любых $s,s'\in S$

$$M(s,\sigma)=M(s',\sigma) \Longrightarrow s=s'$$

Если S является множеством тогда функций, которые отображают S на S взаимооднозначно называются пермутациями. Пермутация f называется регульярной если любая степень f^n равна тождественному отображению множества S для любого $s\in S$ или если f не оставляет произвольного элемента из S фиксирован-

ным ($f^n(s) \neq s$ для любого $s \in S$ и $n = 1,2,\ldots,$). Группа называется семирегуляной если содержит только регулярные пермутации.

Говорим, что группа пермутации P на множестве S является **представлением** группы G если существует отображение группы G на P: $g \longrightarrow \pi(g)$, где $g \in G$ и $\pi(g) \in P$ такие, что

$$\pi(g_1 g_2) = \pi(g_1)\pi(g_2)$$

(группа P является гомоморфичным отображением группы G) .

Если P является изоморфичным отображением группы G ,то представление называется верным. Если любый элемент множества можно отображать на другой произвольный элемент множества S с помощью пермутации из P , тогда группа P называется транзитивной а представление группы G называется транзитивным представлением. Для произвольной подгруппы H группы G , группа G индукует группу пермутации на левых смежных классах под группы H для любого $g \in G$

$$\pi(g) = \begin{pmatrix} Hx \\ Hxg \end{pmatrix} \qquad \text{для } x \in G$$

где Hx является левым смежным классом связанным с элементом x. Если H является подгруппой группы G и $g \longrightarrow \pi(g)$ является транзитивным представлением группы G как группы пермутации левых смежных класс подгруппы H , то элементы подгруппы H , которые отображенны на тождественность группы пермутации, образуют наибольшую нормальную подгруппу N группы содержимой в H и обозначенной N(H) и группа пермутации изоморфна факторгруппе $G/_{N(H)}$. Потому представление является верным тогда и только тогда, когда H не содержит нормальных подгрупп исключая тождественность. Кроме того любое, верное представление группы G как группы пермутации определенно с точностью до изоморфизма через такую подгруппу H , которая не со-

держит нормальных подгрупп исключая тождественность. Когда G является абелевой группой, тогда все верные транзитивные представления группы G как группы пермутации являются изоморфными (потому, что все подгруппы являются нормальными в G) .

Теорема 3. Пусть $A=(S, \Sigma, M)$ будет автоматом и $B=(S, \Sigma, M_h)$ его h -преоброзованием. Автомат A является линейно (обобшенно линейно) реализованным тогда и только тогда, когда B является линейно (обобшенно линейно)реализованным.

Доказательство. Если A является линейно реализованным над GF(p) , то на основании определения действуют отображения α_1, α_2 и матрицы A,B над GF(p) такие, что

$$\alpha_1(M(s,\sigma)) = A\alpha_1(s) + B\alpha_2(\sigma) .$$

Предположим, что α_1 , α_2 являются соответственными отображениями линейной реализации для автомата B. Тогда

$$\alpha_1'(M_h(s,\sigma)) = \alpha_1'(M(s,h(\sigma)) = \alpha_1(M(s,h(\sigma)) = A\alpha_1(s) + B\alpha_2(h(\sigma)).$$

Из этого следует, что $A' = A$, $B' = B$, $\alpha_1' \equiv \alpha_1$, $\alpha_2' \equiv \alpha_2 h$.

Наоборот доказательство является аналогичным и потому оно пропушенно (аналогично для обобщенной линейной реализации) .

Теорема 4. Пусть $A=(S, \Sigma, M)$ будет автоматом и $(ext_r A)^*$ фиксированным аналогом обыкновенного разширения . Автомат является линейно реализованным над GF(p) тогда и только тогда, когда $(ext_r A)^*$ является обобшенно линейно реализован над GF(p) .

Доказательство. Если A линейно реализован над GF(p) , то $(ext_r A)^*$ обобшенно линейно реализован потому, что фиксированный аналог обыкновенного разширения является изоморфным с прямым произведением автомата A и циклического автономического автомата B с периодом T= r .

Наоборот, докажем через отрицание. Пусть A не является линей-

но реализованным автоматом. Покажем, что тоже $(ext_r\ A)^*$ не является обобщенно линейно реализованным автоматом над $GF(p)$. Пусть рассуждим два возможные случаи :

а/ когда А является непермутационным автоматом,

б/ когда А является пермутационным автоматом,

а/ из нашей гипотезы и леммы 1 [8] следует для любого $\sigma_1 \in \Sigma$

$M(s_1, \ _1)=M(s_2, \sigma_1)$ $\qquad$ $M(s_1, \sigma_2)\neq M(s_2, \sigma_2)$

это обозначает, что равенство пар состояний во времени $t+1$ для входного симбола σ_1 не влечет за собой равенства пар состояний во времени $t+1$ для всех входных **символов.** Используя свойства изоморфизмов g_i, $i=0,1,\ldots,r-1$ имеем

$$g_i(M(s_1, \sigma_1))=g_i(M(s_2, \sigma_1)) \Longrightarrow M_i(g_i(s_1), \sigma_1)=M_i(g_i(s_2), \sigma_1)$$

с другой стороны имеем

$$g_i(M(s_1, \sigma_1))\neq g_i(M(s_2, \sigma_2)) \Longrightarrow M_i(g_i(s_1), \sigma_2)\neq M_i(g_i(s_2, \sigma_2)$$

и потому

$$M_i(g_i(s_1), \sigma_1)=M_i(g_i(s_2), \sigma_1) \Longrightarrow M_i(g_i(s_1), \sigma_2)\neq M_i(g_i(s_1), \sigma_2)$$

Потому $(ext_r\ A)^*$ является тоже автоматом не реализованным линейно над $GF(p)$.

б/ Пусть $(ext_r\ A)^*$ будет пермутационным автоматом, который является линейно реализованным над $GF(p)$ (конечно, обыкно - венное разширение пермутационного автомата является пермутационным автоматом). Тогда линейная реализация автомата $(ext_r\ A)^*$ характеризована через подгруппу $K < \overline{I(ext_r\ A)^*}$ (теорема 1 [9] и нормальную коммутативную подгруппу $N \lhd \overline{I(ext_r\ A)^*}$ такую, что $K \cap N = \{id.\}$ и $\Sigma \subseteq N \cdot a$ для любого $a \in \overline{I(ext_r\ A)^*}$ и $n^p=\{id.\}$ для любого $n \in N$. Тогда состояния автомата $(ext_r\ A)^*$ характеризованы через левые смежные классы подгруппы H группы $\overline{I(ext_r\ A)^*}$. Функция переходов M^* я-

вляется определённой через пермутации на множестве левых смежных класс подгруппы H; для любого $\bar{y} \in \overline{I(ext_r\ A)^*}$ имеем

$$M^*(Hx, \bar{y}) = \begin{pmatrix} Hx \\ Hxy \end{pmatrix}$$

для $\bar{x} \in \overline{I(ext_r\ A)^*}$. Автомат A является гомоморфическим отображением автомата $(ext_r\ A)^*$ связанным с характеристическим разбиением с подстановочным свойством $\pi_2 = \{B_0, B_1, \ldots, B_{m-1}\}$

$B_j = \{g_i(s_j) : i=0,1,\ldots,r-1\}$ и $j=0,1,\ldots,m-1$

$S = \{s_0, s_1, \ldots, s_{m-1}\}$. С разбиением π_2 связана подгруппа $H < \overline{I(ext_r\ A)^*}$ такая, что её левые смежные классы относительно группы $\overline{I(ext_r\ A)^*}$ являются классами эквивалентности разбиения π_2. Пусть отображение $\varphi : \pi_2 \longrightarrow \{Hx : \bar{x} \in \overline{I(ext_r A)^*}\}$ будет определено следующим видом

$\varphi(B_i) = Hx$ тогда и **только** тогда, когда $\varphi(M(B_i, y)) = Hxy$ для $\bar{y} \in \overline{I(ext_r\ A)^*}$; $i=0,1,\ldots,m-1$. Это обозначает, что любому блоку разбиения π_2 принадлежит левый смежный класс подгруппы H группы $\overline{I(ext_r\ A)^*}$ тогда и только тогда, когда транзитивные представление группы $\overline{I(ext_r\ A)^*}$ через группу пермутации генерованное подгруппой H содержит пермутацию $Hx \longrightarrow Hxy$ и переход $B_i \xrightarrow{y} M(B_i, y)$. Известно, что $(ext_r\ A)^* = AxB$, где B является точно циклическим автоматом и потому H является циклической подгруппой группы $\overline{I(ext_r\ A)^*}$. Следующая импликация

$$A = Hom(ext_r\ A)^* \Longrightarrow H \supseteq K$$

правильна потому, что $H < \overline{I(ext_r\ A)^*}$ и $K < \overline{I(ext_r\ A)^*}$. Из этого следует, что K является циклической подгруппой. Пусть $H \cap N = N_1$, где N коммутативна подгруппа группы $\overline{I(ext_r\ A)^*}$ связана с линейной реализацией автомата $(ext_r\ A)^*$; является подгруппой для H и для $\overline{I(ext_r\ A)^*}$ и N_1 является циклической подгруппой потому, что все подгруппы циклической группы являются цикличес-

кими. Подгруппа H как циклическая группа является абелевой и из этого следует, что все её подгнуппы нормальные.

Литература

[1] Е.В.Гржимала-Буссе, On periodic representation and reducibility of periodic automata, J. Assoc. Comput. Mach. 16 (1969), 432-441.

[2] Е.В.Гржимала-Буссе, On the automorphisms group of periodic automata extensions, Bulletin de LÁcademie Polonaise des Sciences, Ser. Sci Math. Astr. XXII (1974),3, 325-331.

[3] Е.В.Гржимала-Буссе, On periodic sums and finite automata extensions, Proc. Math. Foundations of Computer Science, Jadwisin n/Warsaw, 1974.

[4] Б.Миколайчак , On the reducibility of periodic automata, Foundations of Control Engineering, vol. 1, N1, 1975.

[5] З.Миадович, Б.Миколайчак, On the automorphisms group of strongly related automata and structural properties of finite automata extensions, Foundations of Control Engineering,vol.1,N2, 1976.

[6] Б.Миколайчак, On some properties of cyclic automata and their extensions, Lecture Notes in Computer Science, vol. 45 ,1976, Math. Foundations of Computer Science,Gdańsk.

[7] Б.Миколайчак, On linear and extended linear realization of generalized finite automata extensions, Computer Science Department, Cornell University,Technical Report,1977.

[8] Ю.Гартманис, В.А.Давис, Homomorphic images of linear sequential machines,J.Comput.System Sciences,1(1967)

[9] Ю.Гартманис, Г.Вальтер, Group-theoretic characterization of linear permutation automata,J.Comput.System Sciences 7,(1973),2,168-189.

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1977

АНТОНИ МИХАЛЬСКИ

О ФУНКЦИОНАЛЬНОЙ ПОЛНОТЕ В ТРЕХЗНАЧНОЙ ЛОГИКЕ СИСТЕМ ФУНКЦИЙ АЛГЕБРЫ ЛОГИКИ ОПИСЫВАЮЩИХ ФУНКЦИОНАЛЬНЫЕ И СОЕДИНЯЮЩИЕ ЭЛЕМЕНТЫ

Институт Вычислительной Техники ПАН, Варшава, ПОЛЬША

Потому, что очень трудно вводить исправления до проекта интегральной схемы большое значение имеют решения, в которых можно модифицировать логическую структуру. Выступает сильное стремление к проектированию "эластичных" логических структур (генерирование модификации на основе универсальной структуры). Более типичными примерами этих решений являются логические схемы реализуемые через постоянную память и программируемые матрицы. Мы работали над общей теорией и методами синтеза комбинационных логических схем построенных с произвольных функциональных элементов с модифицированной структурой соединений. Во время работ над синтезом этих схем были введены модели функциональных и соединяющих элементов в трехзначной логике.

Потому что ограничение класса функций реализуемых через схемы построенные из функциональных и соединяющих элементов может упростить алгоритм синтеза, рассмотрено проблему функциональной полноты для системы трехзначных функций реализуемых через функциональные и соединяющие элементы, при условии, что нет ограничений на способ строения схем.

Рассмотрим функциональный элемент представленный на рис. 1.

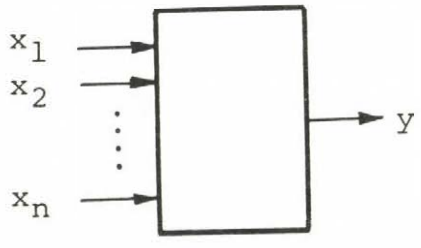


Рис. 1

Предполагаем, что для произвольного числа входов $k, 1 \leq k \leq n$ соединение между источником сигнала управляющего функциональным элементом и входом этого функционального элемента раскрыто. Следовательно предполагаем, что сигнал на выходе функционального элемента тогда определенный и принадлежит к множеству $\{0,1\}$. Эта модель согласна с применяемой обычно в области проверки и диагностики логических схем моделей отказов типа склеивания с 0 по 1. Раскрытие на входе функционального элемента интерпретируется как третье логическое значение 2. Остальные два логические значения обозначаем 0 и 1, как это обычно принято. Область определения функций реализуемых через n-входные функциональные элементы расширяется до $\{0,1,2\}^n$.

Обозначим через $P_2(P_3)$ множество всех булевых (трехзначных) функций. Пусть $F \subset P_2$. Предполагаем, что заданы две функции $f(x_1,x_2,\ldots,x_n)$ и $g(y_1,y_2,\ldots,y_n)$ о областях определения функций D_f и D_g, при этом $D_f \subseteq D_g$. Набор переменных $(x_1,x_2,\ldots,x_n)$ будем обозначать через $\tilde{x}$.

ОПРЕДЕЛЕНИЕ 1

Функцию g назовем расширением функции f (будем писать $f < g$), если для всех $\tilde{x} \in D_f$, $f(\tilde{x}) = g(\tilde{x})$.

ОПРЕДЕЛЕНИЕ 2

Вложением множества функций $F \subseteq P_2$ в P_3 назовем отображение $\phi: F \xrightarrow{b} P_3$, если выполнены следующие условия:

1^0. для всех $f \in F$, $f < \phi(f)$,

2^0. совокупность значений $\phi(f)$ есть множество $\{0,1\}$.

ПРИМЕР 1

Вложение функционального элемента NAND:

a \ b	0	1	2
0	1	1	1
1	1	0	0
2	1	0	0

Соединяющим элементом назовем элемент с двумя входами: информационным p **и управляющем** c **(рис. 2).**

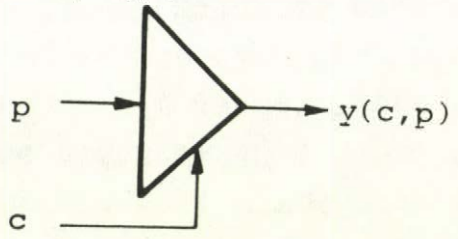


Рис. 2

Значение сигнала на управляющем входе определяет одно из взаимо исключающих условий:

1. **для значений** $p \in \{0,1\}$ **сигнал на выходе** y **есть повторение или инверсия сигнала на входе** p,
2. **выходная импеданция элемента очень большая, то есть на выходе выступает сигнал** 2 **независимо от входа** p.

В работе действие соединяющего элемента описано трехзначной функцией принадлежащей к множеству Q , **определенному далее. Следующие таблицы определяют 2-аргументные функции с областей определения этих функций отсеченной до** $\{0,1\}$**:**

f_1

c \ p	0	1
0	2	2
1	0	1

f_2

c \ p	0	1
0	2	2
1	1	0

f_3

c \ p	0	1
0	0	1
1	2	2

f_4

c \ p	0	1
0	1	0
1	2	2

ОПРЕДЕЛЕНИЕ 3

Функция $g(x_1,x_2)$ **принадлежит множеству** Q , **если**

$$g|\{0,1\}^2 \in \{f_1,f_2,f_3,f_4\}$$

Соединяющие элементы могут иметь разную физическую реализацию например:

- замкнуты или раскрыты соединения,

- тристановые электронные элементы,
- контакты.

ОПРЕДЕЛЕНИЕ 4

Класс $\mathfrak{M}$ функций из P_3 называется функционально замкнутым, если в месте со всякой системой функций $f_1,\ldots,f_s \in P_k$ этому классу принадлежит и любая их суперпозиция.

ОПРЕДЕЛЕНИЕ 5

Система функций из P_3 называется полной в P_3 если каждая функция из P_3 является суперпозицией функций этой системы.

В этой работе рассматриваем следующую проблему:

Есть ли система функций составленная с произвольного вложения системы функций полной в P_2 и произвольной функции соединяющего элемента полна в P_3.

ОПРЕДЕЛЕНИЕ 6

Класс $\mathfrak{N}$ функций принадлежащий P_3 называется предполным в P_3, если $\mathfrak{N}$ представляет неполную в P_3 систему, но присоединение любой функций $f \in P_3$ и $f \notin \mathfrak{N}$ обращает $\mathfrak{N}$ в полную в P_3 систему.

Из определения следует, что предполный в P_3 класс $\mathfrak{N}$ является замкнутым.

Используемая нами теорема о полноте в P_3 была сформулирована Яблонским в 1953 году.

ТЕОРЕМА 1

Для того, чтобы система функций $f_1,\ldots,f_s$ из P_3 была полной необходимо и достаточно, чтобы она не содержалась целиком ни в одном из следующих 18 предполных классов (обозначения как в работе Яблонского):

M_i $(i = 1,2,3)$, $T_{N,2}$, $T_{\varepsilon_{01},0}$, $T_{\varepsilon_{02},0}$, $T_{\varepsilon_{12},0}$, $T_{\varepsilon_0,0}$, $T_{\varepsilon_1,0}$, $T_{\varepsilon_2,0}$,

$T_{\varepsilon_0,1}$, $T_{\varepsilon_1,1}$, $T_{\varepsilon_2,1}$, $U_{\varepsilon_{01},\varepsilon_2}$, $U_{\varepsilon_{02},\varepsilon_1}$, $U_{\varepsilon_{12},\varepsilon_0}$,

L и S_{x+1}.

Используя теорему Яблонского мы доказали следующую теорему.

ТЕОРЕМА 2

Для произвольного вложения системы функций F **полной в** P_2 **и произвольной функции соединяющего элемента осуществлено точно одно из следующих условий:**

1°. $\phi(F) \cup \{g\}$ **полна в** P_3,

2°. $\phi(F) \cup \{g\}$ **принадлежит** $U_{\varepsilon_{0,2},\varepsilon_1}$, **не принадлежит** 17 **остальным предполным классам,**

3°. $\phi(F) \cup \{g\}$ **принадлежит** $U_{\varepsilon_{1,2},\varepsilon_0}$, **не принадлежит остальным** 17 **предполным классам.**

ПРИМЕР 2

Следующая система функций полна в P_3:

ZNAND

x_1 \ x_2	0	1	2
0	1	1	1
1	1	0	0
2	1	0	0

PEM

c \ p	0	1	2
0	2	2	2
1	0	1	2
2	2	2	2

Из Теоремы 2 **вытекают следующие следствия.**

I. **Существуют примеры, когда исследованная система элементов может быть применена до строения троичных комбинационных логических схем.**

II. Алгоритм синтеза схем реализующих булевые функции с модифицированной структурой соединений должен быть универсальный.

III. Основанный на принятых моделях метод синтеза может дать новые решений схем с переменной структурой реализующих булевые функции.

Библиография

Яблонский С. В.: Функциональные построения в к-значной логике, Труды Института им. Стеклова, 51, 1958, стр. 5-142.

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1977

З.Б.МИАДОВИЧ

СИЛЬНАЯ СВЯЗНОСТЬ ПЕРИОДИЧЕСКИХ СУММ КОНЕЧНЫХ АВТОМАТОВ

Познаньский Политехнический Институт

1. Основные понятия и определения

Проблемы, связанные с сильной связностью периодических сумм конечных автоматов, будут рассмотрены в двух случаях. В первом - конечное непустое множество входов есть то же самое для всех автоматов в периодической сумме, во втором - расширено определение периодической суммы так, что множества входов различные.

В этой работе *автоматом* A будем называть тройку (C,Σ,M), где C - конечное, непустое множество состояний, Σ - конечное, непустое множество входов, $M:C\times\Sigma\to C$ - функция следующего состояния.
Автомат есть *сильно связный* тогда и только тогда, если для всяких пар (c,c') состояний в A существует $x\in I$ такое, что $M(c,x)=c'$, где I есть множество всех конечных последовательностей элементов в Σ.
Автомат есть *асинхронный* тогда и только тогда, если для всех $c \in C$ и $\sigma \in \Sigma$ имеем $M(c,\sigma) = M(c,\sigma\sigma)$.
Автомат есть *перестановочный* тогда и только тогда, если для всяких x в I имеем $\{M(c,x): c \in C\} = C$.
Пусть T будет положительным числом. *Точно периодический автомат* /дальше - просто периодический автомат/ V есть тройка (C^+,Σ^+,M^+), где C^+ есть последовательность $C_0^+,C_1^+,\ldots,C_{T-1}^+$ конечных непустых множеств состояний, Σ^+ есть последовательность $\Sigma_0^+,\Sigma_1^+,\ldots,\Sigma_{T-1}^+$ конечных непустых множеств входов, M^+ есть последовательность $M_0^+,M_1^+,\ldots,M_{T-1}^+$ функций следующего состояния, где:
$M_a^+:C_a^+\times\Sigma_a^+\to C_{a+1\,(\mathrm{mod}\,T)}^+$ при $a = 0,1,\ldots,T-1$.
Число T будем называть периодом V.

Закрепленный аналог V* периодического автомата $V=(C^+,\Sigma^+,M^+)$ есть автомат (C^*,Σ^+,M^*), где $C^*=C_0^+ \cup C_1^+ \cup \ldots \cup C_{T-1}^+$, $M^*:C^*\times\Sigma^+\to C^*$

есть функция переходов, определенная для всяких $c \in C^{+}_{a}$, $\sigma \in \Sigma^{+}_{a}$ при $a = 0,1,\ldots,T-1$ следующим образом

$$M^{*}(c,\sigma) = M^{+}_{a}(c,\sigma).$$

Пусть $A^{0} = (C^{0},\Sigma^{0}, M^{0})$, $A^{1} = (C^{1},\Sigma^{1},M^{1}),\ldots, A^{T-1} = (C^{T-1}, \Sigma^{T-1}, M^{T-1})$ будут автоматами.

Пусть $\Psi_{0}:C^{0} \to C^{1}$, $\Psi_{1}:C^{1} \to C^{2},\ldots, \Psi_{T-1}:C^{T-1} \to C^{0}$, $\rho_{0}:\Sigma^{0} \to \Sigma^{1}$, $\rho_{1}:\Sigma^{1} \to \Sigma^{2},\ldots, \rho_{T-1}:\Sigma^{T-1} \to \Sigma_{0}$

будут взаимно однозначными функциями. *Периодическая сумма автоматов* $A^{0},A^{1},\ldots,A^{T-1}$, *связанная с функциями* $\Psi_{0},\ldots,\Psi_{T-1}$, $\rho_{0},\ldots,\rho_{T-1}$, есть периодический автомат $V=(C^{+},\Sigma^{+},M^{+})$ с периодом T, где C^{+} есть последовательность $C^{0},C^{1},\ldots,C^{T-1}$, Σ^{+} есть последовательность $\Sigma^{0}, \Sigma^{1},\ldots, \Sigma^{T-1}$, а M^{+} есть последовательность $M^{+}_{0},M^{+}_{1},\ldots,M^{+}_{T-1}$ где для всех $c \in C^{a}, \sigma_{1} \in \Sigma^{a}$, при $a = 0,1,\ldots, T-1$ имеем

$$\text{(i)} \quad M^{+}_{a}(c,\sigma) = \Psi_{a} M^{a}(c,\sigma)$$

или

$$\text{(ii)} \quad M^{+}_{a}(c,\sigma) = M^{a+I(\mathrm{mod}\ T)}[\Psi_{a}(c), \rho_{a}(\sigma)] .$$

Разумеется, что если конечное непустое множество входов Σ есть то же самое для всех автоматов, то Σ^{+} в этих определениях означает просто конечное непустое множество входов Σ.

Необходимо также заметить, что если $A^{0},A^{1},\ldots,A^{T-1}$ будут автоматами, тогда периодическая сумма этих автоматов, определенная в виде (i), будет различна от периодической суммы, определенной в виде (ii). Это просто доказать с помощью характеристических полугрупп этих двух периодических сумм.

2. Результаты

Теорема 1.

Пусть $A^{0}=(C^{0},\Sigma^{0},M^{0}),\ldots,A^{T-1}=(C^{T-1},\Sigma^{T-1}, M^{T-1})$ будут автоматами.

Пусть $\Psi_{0}:C^{0} \to C^{1},\ldots, \Psi_{T-1}:C^{T-1} \to C^{0}$, $\rho_{0}:\Sigma^{0} \to \Sigma^{1},\ldots,\rho_{T-1}:\Sigma^{T-1} \to \Sigma^{0}$

будут взаимнооднозначными функциями.

Периодическая сумма $V=(C^+,\Sigma^+,M^+)$ автоматов $A^0,\ldots,A^{T-1}$, связанная с функциями $\Psi_0,\ldots,\Psi_{T-1}$, $\rho_0,\ldots,\rho_{T-1}$, определенная как в (i), не является сильно связной тогда и только тогда, если:

$$\{\Psi_a(c): c \in \{M^a(c,\sigma): c \in K^a, \sigma \in \Sigma^a\}\} \subseteq K^{a+1 \pmod T},$$

где K^a есть подмножество множества C^a и $a=0,1,\ldots,T-1$.

Доказательство.

Необходимое условие. Для закрепленного аналога V^* периодической суммы V автоматов $A^0,\ldots,A^{T-1}$, связанной с функциями $\psi_0,\ldots,\Psi_{T-1}$) $\rho_0,\ldots,\rho_{T-1}$, имеем

(v) $\{M^*(c,\sigma): c \in K^a, \sigma \in \Sigma^a\} \subseteq K^{a+1 \pmod T}$

где $a = 0,1,\ldots,T-1$.

Пусть $k \in K^a$ и $t \in C^{a+1} \setminus K^{a+1}$; тогда для пары (k,t) не существует $x \in I$ такое, что $M^*(k,x)=t$. Это значит, что V^* - не сильно связная.

Достаточное условие. Если V^* - не сильно связная, то существуют подмножества $K^0,K^1,\ldots,K^{T-1}$, соответствующие множествам $C^0, C^1,\ldots,C^{T-1}$ такие, что по меньшей мере одно из них является собственным и выполняет условие (v).

Пусть $K^a=C^a$ и $K^{a+1} \subsetneq C^{a+1}$. Тогда $\{M^a(c,\sigma): c \in K^a, \sigma \in \Sigma^a\}=C^a$ потому, что A^a есть сильно связный. Одновременно $\{\Psi_a M^a(c,\sigma): c \in K^a, \sigma \in \Sigma^a\}=C^{a+1}$ потому, что Ψ_a есть функция взаимно однозначная. Тогда мы получили противоречие и последствие должно быть $K^a \subsetneq C^a$ и $K^{a+1} \subsetneq C^{a+1}$.

Теорема 2.

Если периодическая сумма $V=(C^+,\Sigma^+,M^+)$ сильно связных автоматов $A^0,\ldots,A^{T-1}$, связанная с функциями взаимно однозначными $\Psi_0,\ldots,\Psi_{T-1}$, $\rho_0,\ldots,\rho_{T-1}$, определенные как в (ii), то она не будет сильно связной тогда и только тогда, если

$$M^{a+1 \pmod T}[\Psi_a(c), \rho_a(\sigma)] \in \Pi^{a+1 \pmod T}$$

для всяких $c \in \Pi^a$ и $\sigma \in \Sigma^a$, где Π^a есть собственное подмножество множества C^a .

Доказательство.

Необходимое условие. Для закрепленного аналога V^* периодической суммы V автоматов $A^0,\ldots,A^{T-1}$ связанного с функциями $\Psi_0,\ldots,\Psi_{T-1}, \rho_0,\ldots,\rho_{T-1}$, имеем

$$(vv)\quad \{M^*(c,\sigma): c \in \Pi^a , \sigma \in \Sigma^a\} \subseteq \Pi^{a+1 (\mathrm{mod}\ T)}$$

Пусть $\Pi \in \Pi^a$ и $t \in C^{a+1} \setminus \Pi^{a+1}$; тогда для пары (π,t) не существует $x \in I$ такое, что $M^*(\pi,x)=t$. Это означает, что V^* - не сильно связный.

Достаточное условие. Если V^* - не сильно связный, то существуют подмножества $\Pi^0,\ldots,\Pi^{T-I}$ соответствующих множеств $C^0,\ldots,C^{T-I}$ такие, что по меньшей мере одно из них является собственным и выполняется условие (vv) .

Пусть $\Pi^a = C^a$ и $\Pi^{a+1} \subsetneq C^{a+1}$. Тогда $\{M^a(c,\sigma): c \in \Pi^a, \sigma \in \Sigma^a\} = C^a$ потому, что A^0 есть сильно связный. Одновременно,

$$\{M^{a+1 (\mathrm{mod}\ T)}[\Psi_a(c), \rho_a(\sigma)]: c \in \Pi^a, \sigma \in \Sigma^a\} = C^{a+1 (\mathrm{mod}\ T)}$$

потому, что Ψ_a и ρ_a являются взаимно однозначными функциями. Тогда мы получили противоречие и последствие должно быть $\Pi^a \subsetneq C^a$ и $\Pi^{a+I} \subsetneq C^{a+1}$.

Теорема 3.

Пусть $A^0 = (C^0, \Sigma^0, M^0) ,\ldots, A^{T-1} = (C^{T-1}, \Sigma^{T-1}, M^{T-1})$ будут сильно связными асинхронными автоматами.

Пусть $\Psi_0: C^0 \to C^I,\ldots,\Psi_{T-1}: C^{T-1} \to C^0$, $\rho_0: \Sigma^0 \to \Sigma^1,\ldots,\rho_{T-1}: \Sigma^{T-1} \to \Sigma^0$

будут взаимно однозначными функциями. Тогда закрепленный аналог V^* периодической суммы V автоматов $A^0,\ldots,A^{T-1}$, связанный с функциями $\Psi_0,\ldots,\Psi_{T-1},\rho_0,\ldots, \rho_{T-1}$, есть сильно связный.

Доказательство.

1. Пусть периодическая сумма будет определена в виде (i) .

Пусть $K^1,\ldots,K^{T-1}$ будут собственными подмножествами соответствующих множеств $C^0,\ldots,C^{T-1}$. Из определения асинхронного автомата имеем

(iv) $\quad |\{M^a(c,\sigma): c \in K^a,\ \sigma \in \Sigma^a\}| > |K^a|$

Это значит, что не существуют собственные подмножества $K^0,\ldots,K^{T-1}$, выполняющие условия

$$\{\Psi_a(c): c \in \{M^a(c,\sigma): c \in K^a,\ \sigma \in \Sigma^a\}\} \subseteq K^{a+1(\mathrm{mod}\ T)}$$

потому, что как вытекает из (iv)

$$|\{\Psi_a(c): c \in \{M^a(c,\sigma): c \in K^a,\ \sigma \in \Sigma^a\}\}| > |K^a|$$

а Ψ_a есть взаимно однозначная функция. Тогда мы получили противоречие такое, что $|K^0| > |K^0|$.

2. Пусть периодическая сумма будет определенной в виде (ii).

Пусть $\Pi^0,\ldots,\Pi^{T-1}$ будут собственными подмножествами соответствующих множеств $C^0,\ldots,C^{T-1}$. Из определения асинхронного автомата имеем

(ivv) $\quad |\{M^a(c,\sigma): c \in \Pi^a,\ \sigma \in \Sigma^a\}| > |\Pi^a|.$

Пусть $\pi \in \Pi^a$ и $\tau \in C^{a+1}\setminus\Pi^{a+1}$; тогда для пары (π,t) существует $x \in I$ такое, что $M^*(\pi,x)=t$ потому, что в каждом из автоматов $A^0,\ldots,A^{T-1}$ имеем

$$\sum_{a=0}^{T-1} |\{M^{a+1(\mathrm{mod}\ T)}[\Psi_a(c),\rho_a(\sigma)]: c \in \Pi^a, \sigma \in \Sigma^a\}| > |\Pi^a|$$

так как Ψ_a и ρ_a - взаимно однозначные. Тогда мы получили противоречие такое, что $|\Pi^a| > |\Pi^a|$.

Теорема 4.

Пусть $A^0=(C^0,\Sigma^0,M^0)$, ..., $A^{T-1}=(C^{T-1},\Sigma^{T-1},M^{T-1})$ будут сильно связными пермутационными автоматами.

Пусть $\Psi_0: C^0 \to C^1,\ldots, \Psi_{T-1}: C^{T-1} \to C^0, \rho_0: \Sigma^0 \to \Sigma^1,\ldots, \rho_{T-1}: \Sigma^{T-1} \to \Sigma^0$

будут взаимно однозначными функциями.

Пусть будет подмножество T^M множества C^M такое, что

$$\{M^M(c,\sigma_a): c \in T^M, \sigma_a \in \Sigma^M\} \neq \{M^M(c,\sigma): c \in T^M, \sigma \in \Sigma^M\},$$, где

$M \in \{0,1,\ldots,T-1\}$. Тогда закрепленный аналог периодической суммы автоматов $A^0,\ldots,A^{T-1}$, связанный с функциями $\Psi_0,\ldots,$ $\Psi_{T-1},\rho_0,\ldots,\rho_{T-1}$ есть сильно связный.

Доказательство.

1. Пусть периодическая сумма будет определенной в виде /i/.

Пусть V^* не есть сильно связный.

Пусть $K^0,\ldots,T^M,\ldots,K^{T-1}$ будут подмножествами соответственно множествам $C^0,\ldots,C^M,\ldots,C^{T-1}$ такими, что

$$\{\Psi_a(c): c \in \{M^a(c,\sigma): c \in K^a, \sigma \in \Sigma^a\}\} \subseteq K^{a+1(\mathrm{mod}\ T)}$$

где $a=0,\ldots,M,\ldots,T-1$. Из определения пермутационного автомата имеем

$$|\{\Psi_a(c): c \in \{M^a(c,\sigma): c \in K^a, \sigma \in \Sigma^a\}\}| \geq |K^a|,$$

где $a = 0,\ldots,M-1, M+1,\ldots,T-1$ и

$$|\{\Psi_M(c): c \in \{M^M(c,\sigma): c \in T^M, \sigma \in \Sigma^M\}\} > |T^M|.$$

Это значит, что $|K^{a+1}|$ должно быть равным $|K^a|$. Тогда мы получили противоречие такое, что $|K^{M+I}| > |T^M|$.

2. Пусть периодическая сумма будет определенной в виде (ii) .

Пусть $\Pi^0,\ldots,T^M,\ldots,\Pi^{T-1}$ будут собственными подмножествами соответственно множествам $C^0,\ldots,C^M,\ldots,C^{T-1}$.

Из условия о не сильной связности периодической суммы /Теорема 2/, а также из определения пермутационного автомата вытекает, что должно быть $|\Pi^0|=\ldots=|\Pi^{M-1}|=|T^M|=|\Pi^{M+1}| = \ldots = |\Pi^{T-1}|$; но из предположения о подмножестве T^M имеем, что существует $c \in \Pi^{M-1}$, $\sigma \in \Sigma^{M-1}$ такое, что $M^M[\Psi_{M-1}(c),\rho_{M-1}(\sigma)] \notin T^M$ потому, что Ψ_{M-1} и ρ_{M-1} являются взаимно однозначными функциями.

Теорема 5.

Пусть $A^0,\ldots,A^{T-1}$ будут сильно связными асинхронными или перестановочными автоматами.

Пусть множество всех асинхронных автоматов множества $A^0,\ldots,A^{T-1}$ непустое. Пусть $\Psi_0,\ldots,\Psi_{T-1}$, $\rho_0,\ldots,\rho_{T-1}$ будут взаимно однозначными функциями. Тогда закрепленный аналог периодической суммы автоматов $A^0,\ldots,A^{T-1}$, связанный с функциями $\Psi_0,\ldots,\Psi_{T-1},\rho_0,\ldots,\rho_{T-1}$, есть сильно связный.

Доказательство.

Просто вытекает из доказательств Теоремы 3 и Теоремы 4.

Теорема 6.

Пусть $A^0,\ldots,A^{T-1}$ будут сильно связными автоматами.

Пусть $\Psi_0,\ldots,\Psi_{T-1},\rho_0,\ldots,\rho_{T-1}$ будут взаимно однозначными функциями. Пусть все $A^{i_1},A^{i_2},\ldots,A^{i_r}$ будут асинхронными автоматами, выбранными из $A^0,\ldots,A^{T-1}$, пусть $A^{i_{r+1}}, A^{i_{r+2}}, \ldots, A^{i_T}$ будут множеством всех неасинхронных автоматов из множества $A,\ldots,A^{T-1}$. Пусть $T^{i_{r+1}}, T^{i_{r+2}},\ldots,T^{i_T}$ будут собственными подмножествами соответственно множествам $C^{i_{r+1}},C^{i_{r+2}},\ldots,C^{i_T}$ такими, что для всяких $a=r+1, r+2,\ldots,T$ число $|T^{i_a}|-|\{M^{i_a}(c,\sigma): c \in T^{i_a},\sigma\in\Sigma^{i_a}\}|$ есть максимальное и положительное.

Пусть $\pi = \sum_{a=r+1}^{T} |T^{i_a}|$ $\quad T = \sum_{a=r+1}^{T} |\{M^{i_a}(c,\sigma): c \in T^{i_a},\ \sigma\in\Sigma^{i_a}\}|$.

Тогда закрепленный аналог периодической суммы автоматов $A^0,\ldots,A^{T-1}$, связанный с функциями $\Psi_0,\ldots,\Psi_{T-1}$, $\rho_0,\ldots,\rho_{T-1}$, есть сильно связный, если неравенство $r>\pi-t$ удовлетворено.

Доказательство.

Из Теоремы 1 и Теоремы 2 вытекает, что периодическая сумма не есть сильно связная, если $\sum_{M-1}^{r} |T^{i_M}|$ для асинхронных автоматов не меньше, чем число t неасинхронных автоматов; тогда $\sum_{M=1}^{r} |T^{i_M}| \geq t$. Следовательно, $\sum_{M=1}^{r} |\{M^{i_M}(c,\sigma): c\in T^{i_M}, \sigma\in\Sigma^{i_M}\}|$ для асинхронных автоматов есть не меньше, чем $t+r$ потому, что

как вытекает из определения асинхронного автомата:

$$|\{M(c,\sigma): c \in T, \sigma \in \Sigma\}| \geq |T| + 1.$$

Согласно Теореме 1 и Теореме 2 закрепленный аналог периодической суммы автоматов $A^{0},\ldots,A^{T-1}$ не есть сильно связный тогда и только тогда, если $t+r < \pi$ или $r < \pi-t$. Тогда мы получили противоречие.

Цитированная литература.

1. J.W.Grzymala-Busse - "On the connectivity of the periodic sum of automata". Math.Found. of Comp.Sci., High Tatras, Sept.3-8, 1973.

2. J.W.Grzymala-Busse - "On the periodic sum and extensions of finite automata". Proc. in Sommer School of Math.Found. of Computer Sci., Jadwisin 1974.

3. Z.B.Miadowicz, B.Mikolajczak - "On the connectivity of the periodic sum of finite automata". Found. of Contr.Eng., Vol. 1, No 2, 1976.

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1977

IVICS J., SZLÁVIK L.

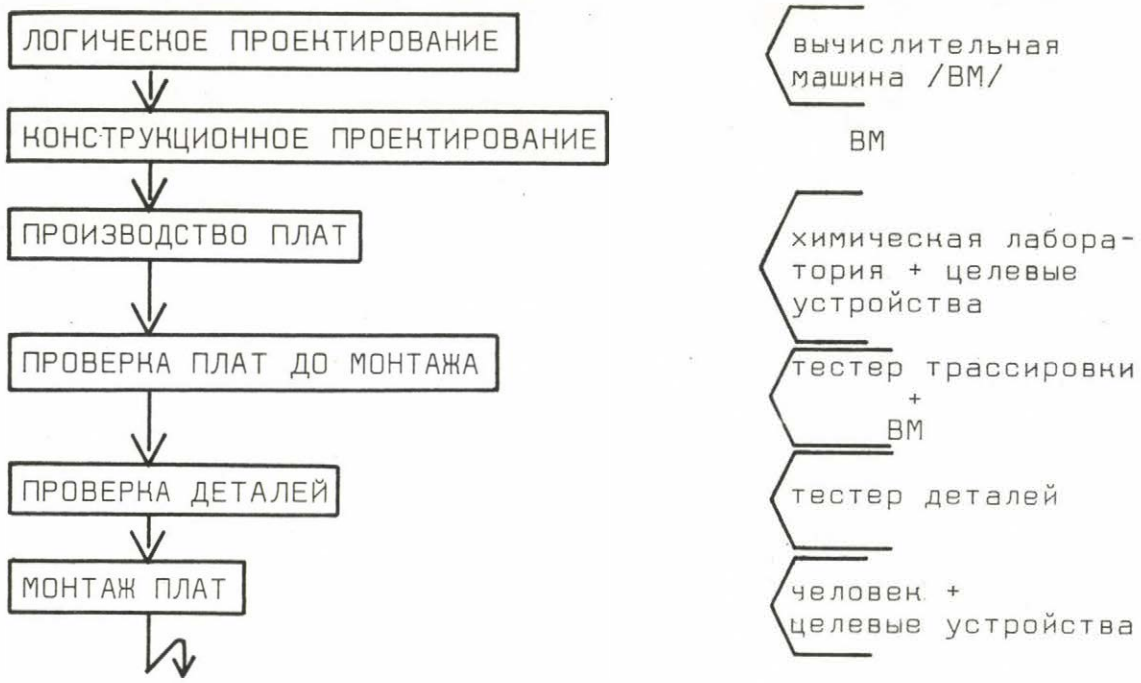
НЕКОТОРЫЕ ВОПРОСЫ ПРОВЕРКИ СМОНТИРОВАННЫХ ПЕЧАТНЫХ ПЛАТ

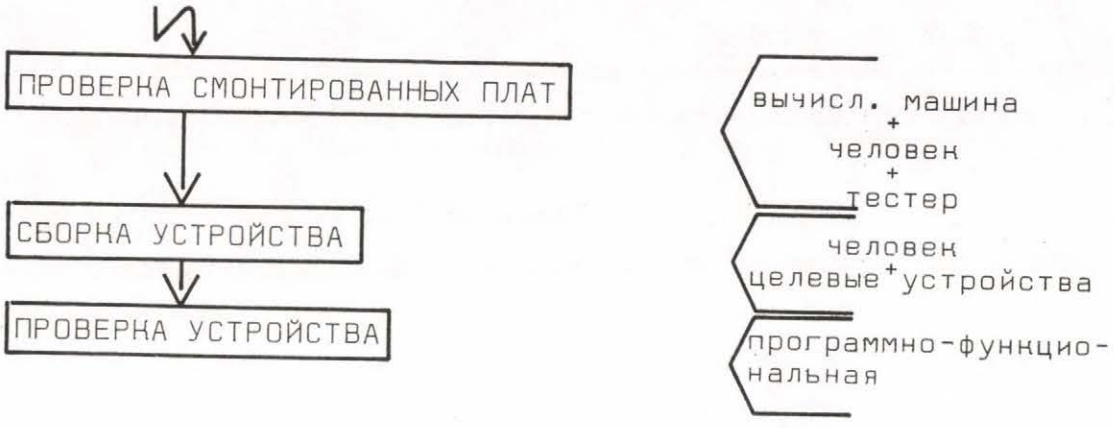
Исследовательский институт
вычислительной техники и автоматизации ВАН

ВВЕДЕНИЕ

В этот раз нам бы хотелось ознакомить Вас с построением, средствами осуществления и местом в автоматическом проектировании разрабатываемой отделом цифровой техники ИИВТиА ВАН системы для автоматической генерации обнаруживающих ошибку тестов смонтированных печатных плат, содержащих цифровые схемы, и на основании сгенерированных тестов - автоматическую локализацию ошибок.

1. АВТОМАТИЧЕСКАЯ ПРОЕКТИРУЮЩАЯ, ПРОИЗВОДЯЩАЯ И ПРОВЕРЯЮЩАЯ СИСТЕМА цифровых устройств при помощи вычислительных машин





Из приведенных процессов для логического проектирования /от логического определения задания до изготовления логического проекта/ и конструкционного проектирования /разделение логического проекта на блоки, платы и проектирование трассировки/ в общем требуется мощная вычислительная машина. Изготовление плат, проверка плат до монтажа, проверка деталей и монтаж плат требуют выработку управляющих лент, применяемых целевым устройством /с помощью так называемых "постпроцессорных программ"/, что может выполняться как на малой, так и на большой вычислительной машине. Подробно мы будем заниматься проверкой смонтированных плат.

2. ПРОВЕРКА СМОНТИРОВАННЫХ ПЛАТ.

Проверка смонтированных плат, в основном, означает решение двух задач:

первая: выработка тестов, необходимых для проверки;

вторая: выполнение тестов, необходимых для проверки.

Способ решения обоих задач определяется:

- числом и/или сложностью деталей, находящихся на платах /100-150 корпусов средней степени интеграции или соответствующая им сложность/;
- возможностями устройства для выполнения тестов /TESTOMAT C/;
- желаемой диагностической разрешающей способностью тестов /локализация или только обнаружение ошибки/.

2.1. ВЫРАБОТКА ТЕСТОВ.

Для выработки тестов сетей размерностью от 100 до 5000 вентильных эквивалентов традиционные методы генерации тестов / D-алгоритм, булева производная, эквивалентная нормальная форма и т.д./ применимы, но для сетей больших размерностей они уже оказываются не пригодными. Выходом для них могут быть: выработка функциональных тестов или вероятностные методы генерации тестов, или каким-то способом разбиение сети на квази-независимые подсети и, составляя для них тесты, составляются тесты и для всей сети. Очевидно, что любой из перечисленных методов требует обработку большого количества данных; кроме того, проверка справедливости и полноты полученных тестов также является частью выработки тестов, в связи с чем для решения задания выработки тестов требуется мощная и быстродействующая вычислительная машина.

Обнаруживающей /ошибку/ тестпрограммой сети называем ту тестпрограмму, которая только обнаруживает наличие ошибки сети и при этом она не определяет ни конкретную причину, ни - соответственно - место ошибки /то-есть, выработка оценки: годна или не годна сеть с теста на тест/. Целью ATER является создание такой логической входной последовательности, которая полностью "*обходит*" граф состояний автомата, изображающего моделированную "*исправную сеть*". Целевая задача обхода - нахождение локального минимума "*полного пути*".

На следующем рисунке показано построение системы для выработки тестов ATER на мощной вычислительной машине типа CDC 3300 или ЕС 1050:

Описание сети - может быть задано в виде геометрическо-технологического или логическо-топологического описания сети.

Модели ошибок - могут быть взяты из каталога или заданы пользователем, указывая подсеть для которой определены.

Описание задания - содержит информацию о том, какие тесты должны вырабатываться: обнаруживающие или локализирующие;

проверить полноту и правильность заданных тестов;
анализировать, а затем перерабатывать исходную сеть;
системные управляющие информации.

Программа-анализатор - осуществляет разбиение сети по потребностям алгоритмических генераторов тестов;
выделяет повторяющиеся, разветвляющиеся и избыточные части сети.
Результатом ее является переработанная схема сети.

Каталог № 1 содержит:

- транслятор с входного языка на внутрисистемный,
- каталог интегральных схем /логическое, конструктивное, технологическое построение, особенности/,
- систему управления базой данных.

Результаты анализа - сообщение о работе транслятора, о результатах программы анализа, об ошибках в описании сети, задания и модели ошибок.

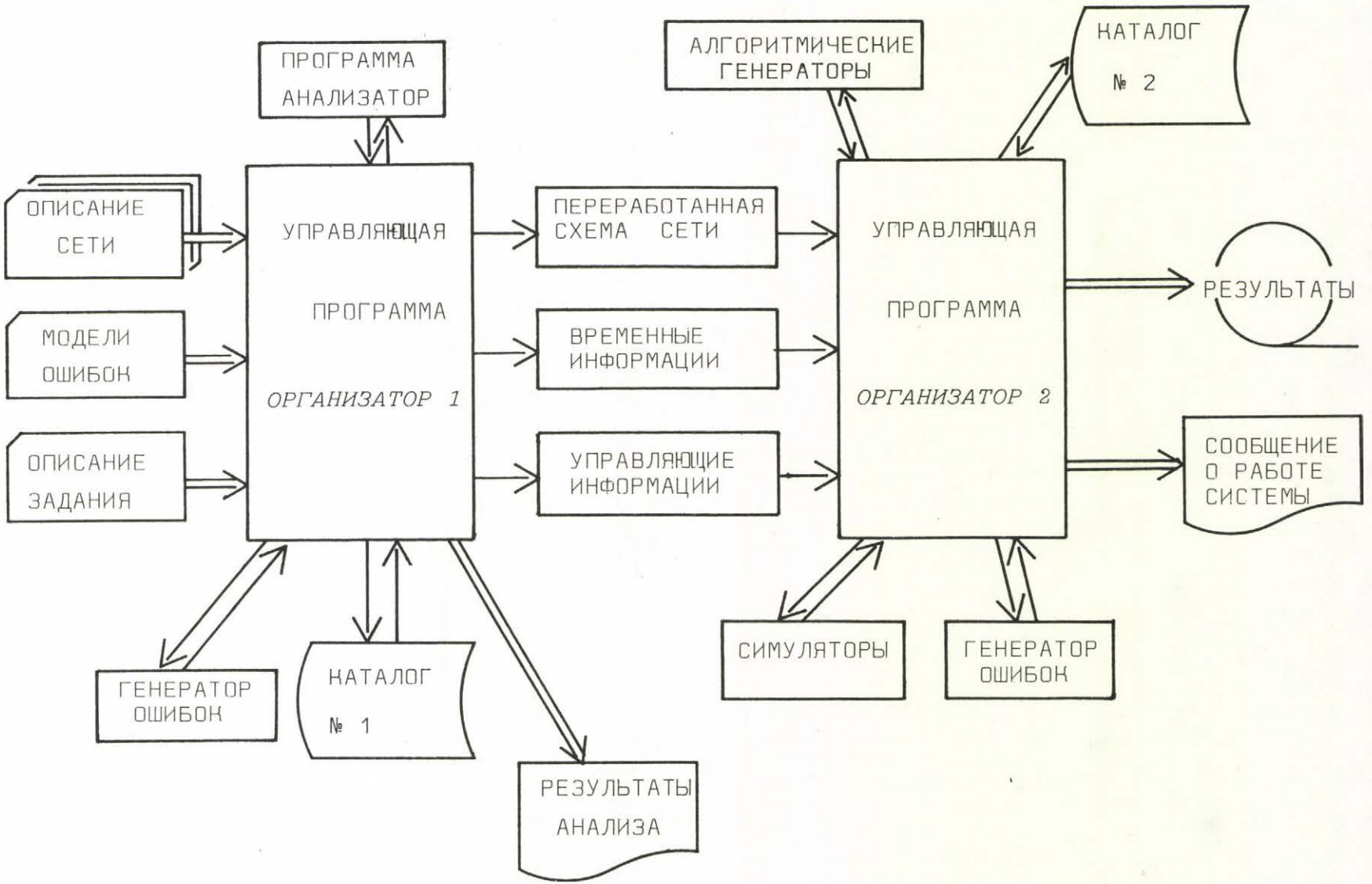
Алгоритмические генераторы тестов по описанию задания вырабатывают с использованием симуляторов обнаруживающие или локализирующие тесты.

Каталог № 2 содержит:

- ошибки, заданные пользователем,
- систему управления базой данных,
- каталог интегральных схем,
- трансляторы, переводящие готовые тесты на язык описания тестов,
- постпроцессор для выработки управляющей ленты устройства TESTOMAT C.

Симуляторы исправных /для вычисления ответа/ и неисправных /для проверки полноты и правильности тестов/ сетей.

Генератор ошибок - при автоматической выработки тестов и при симуляции обеспечит внесение ошибок в сеть, а при пользовательских ошибках обеспечит соответствующее управление ими.


ПРОГРАММА
АНАЛИЗАТОР
ОПИСАНИЕ
СЕТИ
МОДЕЛИ
ОШИБОК
ОПИСАНИЕ
ЗАДАНИЯ
УПРАВЛЯЮЩАЯ
ПРОГРАММА
ОРГАНИЗАТОР 1
ГЕНЕРАТОР
ОШИБОК
КАТАЛОГ
№ 1
РЕЗУЛЬТАТЫ
АНАЛИЗА
ПЕРЕРАБОТАННАЯ
СХЕМА СЕТИ
ВРЕМЕННЫЕ
ИНФОРМАЦИИ
УПРАВЛЯЮЩИЕ
ИНФОРМАЦИИ
АЛГОРИТМИЧЕСКИЕ
ГЕНЕРАТОРЫ
КАТАЛОГ
№ 2
УПРАВЛЯЮЩАЯ
ПРОГРАММА
ОРГАНИЗАТОР 2
РЕЗУЛЬТАТЫ
СООБЩЕНИЕ
О РАБОТЕ
СИСТЕМЫ
СИМУЛЯТОРЫ
ГЕНЕРАТОР
ОШИБОК

Результаты - их вид зависит от желания пользователя и может использоваться либо непосредственно для управления тестера, либо в качестве входных данных системы выполнения тестов ATVR.

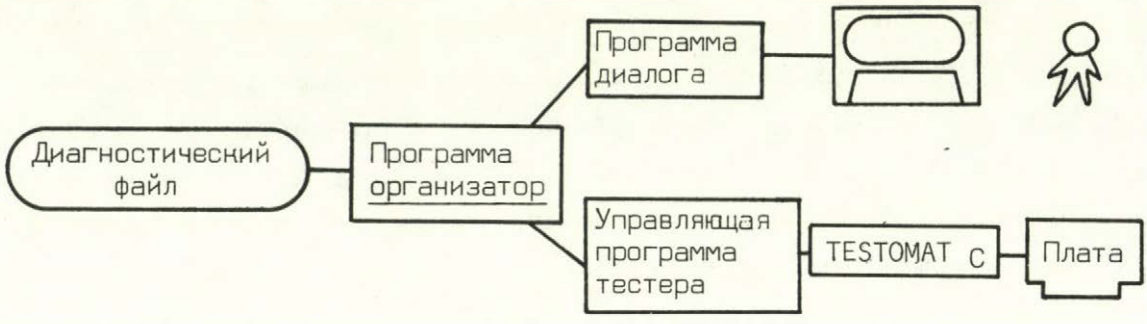
Сообщение о работе системы - результат работы генераторов теста, симуляторов, список необнаруживаемых ошибок.

2.2. ВЫПОЛНЕНИЕ ТЕСТОВ.

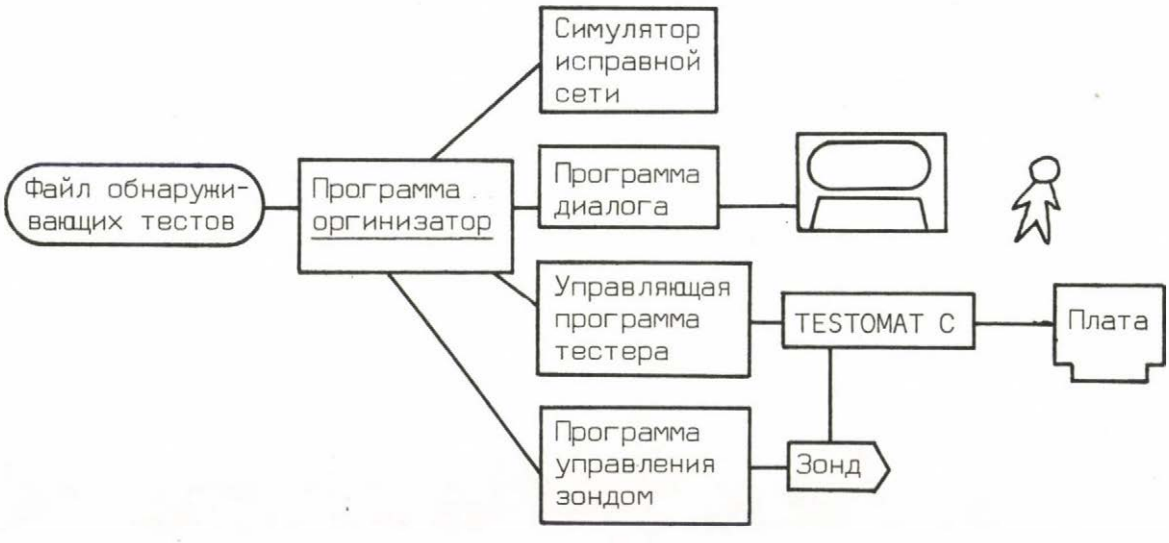
Выполнение тестов состоит из процессов обнаружения ошибок и следующей за ним локализации ошибок. Оба процесса требуют использования вычислительной /малой/ машины. При обнаружении ошибок, то-есть годна или не годна плата, она устанавливается из ответов, измеренных на /первичных/ выходных клеммах, выработанных в результате поданного возбуждения на /первичные/ входные клеммы сети. Локализация ошибок может решаться трояко; все три метода исполним независимо или один за другим.

Процесс выполнения тестов для плат

Локализация ошибок *автоматическая*, если при выработке тестов и обнаруживающие, и локализирующие последовательности вырабатываются и они переводятся вместе с информациями, необходимыми для управления тестера, в управляющую ленту тестера. Этот способ, в первую очередь, целесообразно применять для плат, произведенных в больших сериях,

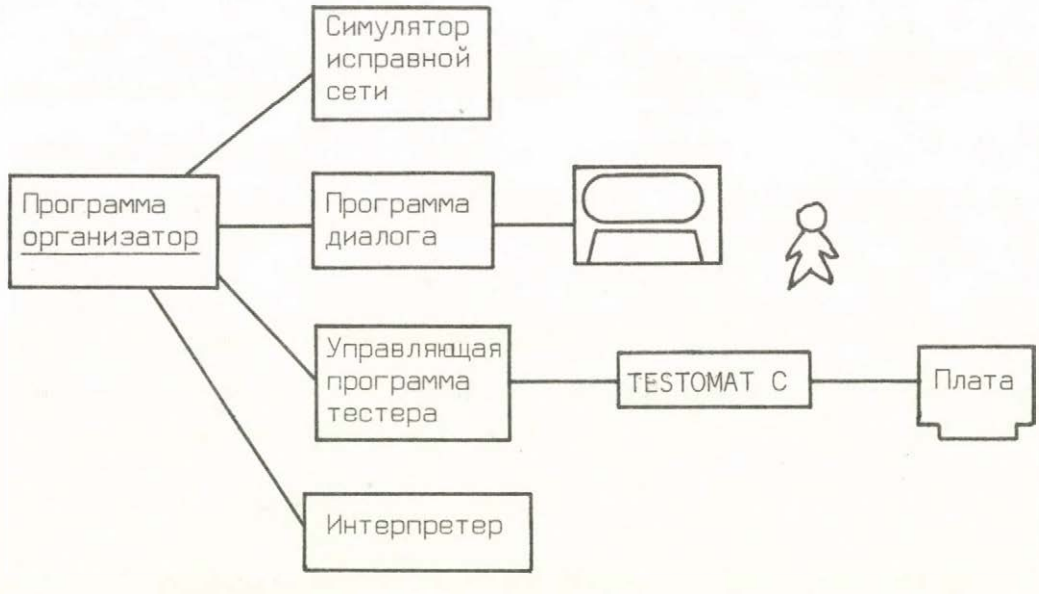


но после достижения определенной сложности платы применение бумажной управляющей ленты станет затруднительным. Поэтому необходимо использовать другой внешний носитель данных, например, - магнитную ленту или магнитный диск. При этом возникает идея передавать только обнаруживающие тесты в вычислительную машину, управляющую выполнением тестов, и вырабатывать на ней необходимую информацию для локализации ошибок. В процессе локализации это означает следующее: если измерения производятся только на /первичных или/ внешних клеммах, то становятся необходимыми генерация и обработка большого количества данных, которые превысят возможности малой машины; поэтому компромиссным решением станет *применение зонда ИС*.



Таким образом станет возможным измерение на внутренних точках платы, что приводит к существенному уменьшению одновременно необходимой информации для локализации так, чтобы на основании результатов измерений программа управления зондом определяла следующее положение зонда до тех пор, пока не выявлен неисправный элемент /корпус ИС/. Преимуществом данного способа является то, что хотя и требуется вмешательство человека, все же класс локализируемых ошибок будет существенно шире учтенных при выработке обнаруживающих тестов.

При локализации ошибок управляемый *человеком* процесс локализации определяется человеком, вычислительная машина выполняет поданные команды и выдает сообщение. Здесь "человек" высококвалифицированный специалист. Этот способ применяется для одиночных плат или если после выполнения 2-х вышеописанных способов на плате остались нелокализированные ошибки.



Локализация ошибки, управляемая человеком

3. СВЯЗЬ ВЫРАБОТКИ ТЕСТОВ И ЛОКАЛИЗАЦИИ ОШИБОК.

Как было видно в предыдущих разделах, автоматическая проверка цифровых смонтированных печатных плат осуществляется двумя органически связанными системами - системой выработки и системой выполнения тестов. Конечной целью проверки является - кроме обнаружения возможно появляющейся ошибки - определение и места /причины/ ошибки на плате, т.е. локализация до самого маленького заменяемого или ремонтируемого элемента. Другими словами, указать неисправный элемент /ИС/ или ошибки трассировки платы.

В процессе автоматической проверки возможности локализации ошибок определяются спецификацией используемого автоматического тестера /TESTOMAT C + управляемый зонд/. Алгоритм локализации ошибок вместе с применяемыми моделями ошибок и способом измерения определяют параметры тест-программы /программа испытаний/, выданной системой выработки тестов ATER.

Важной особенностью системы выполнения тестов ATVR является то, что она должна быть способна самостоятельно локализировать ошибки при применении тест-программы, проектируемой человеком с низкой диагностической разрешающей способностью.

Выработанная ATER тест-программа обеспечивает обнаружение возможно появляющейся ошибки на первичных выходных клеммах сети с помощью устройства TESTOMAT C.

Тест-программа T состоит из последовательности тестов t_i , где $i = 1, \ldots, n$. Состав одного теста: входное возбуждение, соответствующий ему выходной ответ, способ выполнения, а также общие информации для выполнения тестером.

На основе вышеприведенного и определения тестпрограммы система ATVR способна для локализации следующих ошибок:

1/. *Разрыв сигнального пути*

Имеется разрыв сигнального пути, если существует тест $t_i \in \{T\}$, при котором логическое значение двух его узлов разное.

2/. Короткое замыкание сигнальных путей P

Говорим о коротком замыкании между двумя сигнальными путями p_j и p_k, если для всех тестов $t_i \in \{T\}$ значения p_j и p_k одинаковы ($p_j^o \equiv p_k^o$) и существует тест $t_i \in \{T\}$, при котором $p_j^o \cup p_k^o$ оказываются ошибочными. Очевидно, не может быть короткого замыкания между p_j и p_k, если существует тест $t_i \in \{T\}$, при котором $p_j^o \neq p_k^o$.

3/. Постоянство значения сигнального пути

По сигнальному пути $p_j \in \{P\}$ есть ошибка типа постоянство значения, если для всех $t_i \in \{T\}$, p_j имеет постоянное значение $C \in \{0,1\}$ ($p_j^o = C$) и существует $t_i \in \{T\}$, при котором значение C ошибочное.

4/. Функциональная ошибка детали

Деталь IC_k является ошибочной /неисправной/, если существует тест $t_i \in \{T\}$, при котором она не выполняет свою функцию под действием поданных входных последовательностей, и не имеются при этом ошибки типа разрыв, короткое замыкание и постоянство значения в связанном с деталью сигнальном пути.

Эти четыре типа ошибок составляют модель ошибки, которая особенно из-за второго и четвертого типов способна покрыть практически все ошибки смонтированных цифровых печатных плат.

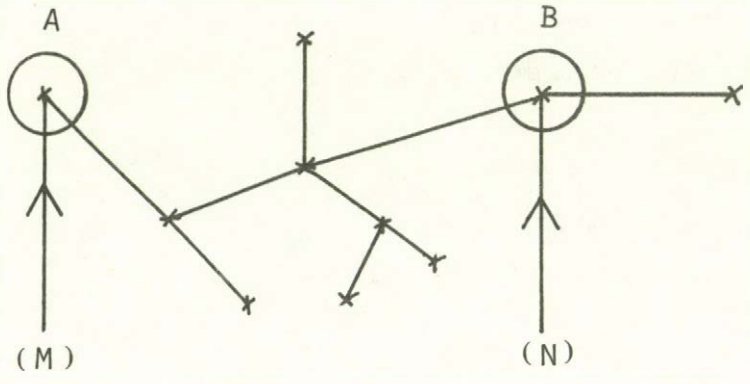
4. АЛГОРИТМЫ ЛОКАЛИЗАЦИИ ОШИБОК

Пусть предположим, что имеется полный набор обнаруживающих тестов сети. В общем-то считается важной временная последовательность /последовательностность/ отдельных тестов. Поэтому, тесты в последовательности идентифицируем их порядковыми номерами */тестномера/*.

4.1. СЛУЧАЙ РАЗРЫВА

Если имеются два узла сигнального пути, которые при разных тестномерах оказываются в первый раз ошибочными, то-есть логи-

ческое значение двух узлов неодинаково для всех тестномеров, то в соответствии с определением /1/ мы говорим о разрыве сигнального пути. /Если не был бы разрыв, то при том же тестномере должны были бы оказаться ошибочными/



где:

A,B - два узла одного и того же сигнального пути;

M - тестномер, при котором в первый раз выявлена ошибка в узле A;

N - тестномер, при котором в первый раз выявлена ошибка в узле B.

При $M \neq N$ имеется разрыв. /Конечно может быть, что один из узлов никогда не оказывается ошибочным./

4.2. СЛУЧАЙ КОРОТКОГО ЗАМЫКАНИЯ

Для снижения числа операций поиска и вычислений целесообразно проверять сигнальные пути, находящиеся в топологической окрестности исследуемого сигнального пути. /Мало вероятно короткое замыкание между физически далекими сигнальными путями./ Возможно, исследуемые случаи определяются всегда технологическими особенностями. Отношение соседства сигнального пути: говорят, что два сигнальных пути - соседние, если имеют хотя бы две соседних точки.

Так как критерий короткого замыкания основан на равенстве логических значений независимых сигнальных путей, поэтому, *если любые две точки имеют разные значения* хотя бы при одном тестномере, то между ними *невозможно* короткое замыкание.

4.3. СЛУЧАЙ ПОСТОЯНСТВА ЗНАЧЕНИЯ

Если сигнальный путь сети имеет *при всех тестах* одинаковое логическое значение и существует тест, при котором оказывается сигнальный путь ошибочным, мы предположим ошибку типа постоянства значения, если:

- нет распространения сигнала в сети, и
- были уже проверены стоки сигнального пути и оказывались исправными, и
- подсеть всегда имеет одно и то же значение.

Характерные виды ошибки постоянства значения:
короткое замыкание сигнального пути с путем питания и сигнальных путей, когда значение одного из них является определяющим.

4.4. НЕИСПРАВНОСТИ ДЕТАЛЕЙ И ФУНКЦИОНАЛЬНЫХ БЛОКОВ

Локализация их предполагает, что в сети уже все вышеописанные виды локализаций были проведены и, при этом, не оказались ошибочными.

З А К Л Ю Ч Е Н И Е

В данной статье мы пытались в общих чертах ознакомить читателя с системными и моделестроящими проблемами проверки смонтированных цифровых печатных плат без претензии на полноту освещения. Мы надеемся, что после практического осуществления системы удастся устранить и этот недостаток.

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1977

K. SAPIECHA

ДИАГНОСТИЧЕСКИЕ МОДЕЛИ АСИНХРОННЫХ КОНЕЧНЫХ АВТОМАТОВ

Институт Вычислительной Техники
ВТУ Варшава

1. ВВЕДЕНИЕ

Определение тестов неисправностей типа "постоянное логическое значение" в асинхронных автоматах состоит в решении двух проблем. Первая из них касается определения структуры интерационной комбинационной схемы, моделирующей действие асинхронного автомата. Вторая требует установления правил определения тестов неисправностей, отображенных в вышеупомянутой модели.

Чаще всего применяемые решения первой из этих проблем, используют модель, введенную Путзолу и Ротом [1]. Здесь будут представлены предложения, касающиеся конструирования правильных моделей [5].

Наиболее распространенным методом решения второй проблемы является так называемая алгебра Рота [2]. В статьи будут введены две других алгебры.

Первая, трехзначная послужит для описания некоторого подкласса схем, вторая, являющаяся расширением алгебры Рота, даст возможность более эффективного определения тестов для любых переключательных схем.

2. ИНТЕРАЦИОННАЯ МОДЕЛЬ АСИНХРОННОГО АВТОМАТА

Будут рассмотрены асинхронные автоматы, рис. 1. Аналогично [1] будем пользоваться гипотезой о возможности размыкания линий обратной связи и составления интерационной комбинационной модели, представляющей асинхронную схему. Принципы составления та-

кой модели коренным образом влияют на качество получаемых тестов. Неправильная модель может привести к результатам не соответствующим принципам действия физических схем.

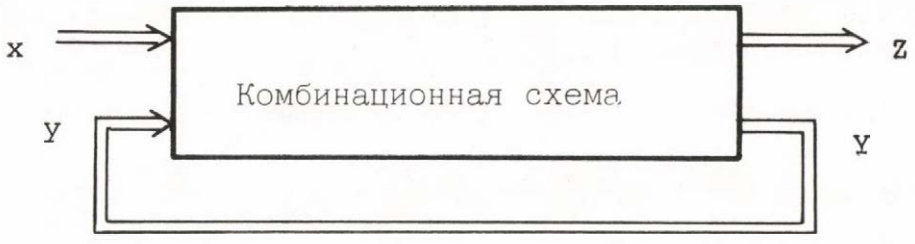


Рис. 1. Асинхронный автомат

С целью проиллюстрировать проблему приведем примеры трех ее различных решений. На рис. 2а показан типичный триггер RS. "Разрез" минимального количества обратных связей ведет к модели, показанной на рис. 2б. Метод использования синхронного варианта схемы [1] обеспечивает результат, представленный на рис. 2в. Соблюдение условия стабильности вынужденного состояния схемы /согласно действию его физического осуществления/ может быть достигнуто при помощи модели, показанной на рис. 2д. Нетрудно доказать, что эта модель наиболее полным образом соответствует требованиям верной репрезентации физической схемы.

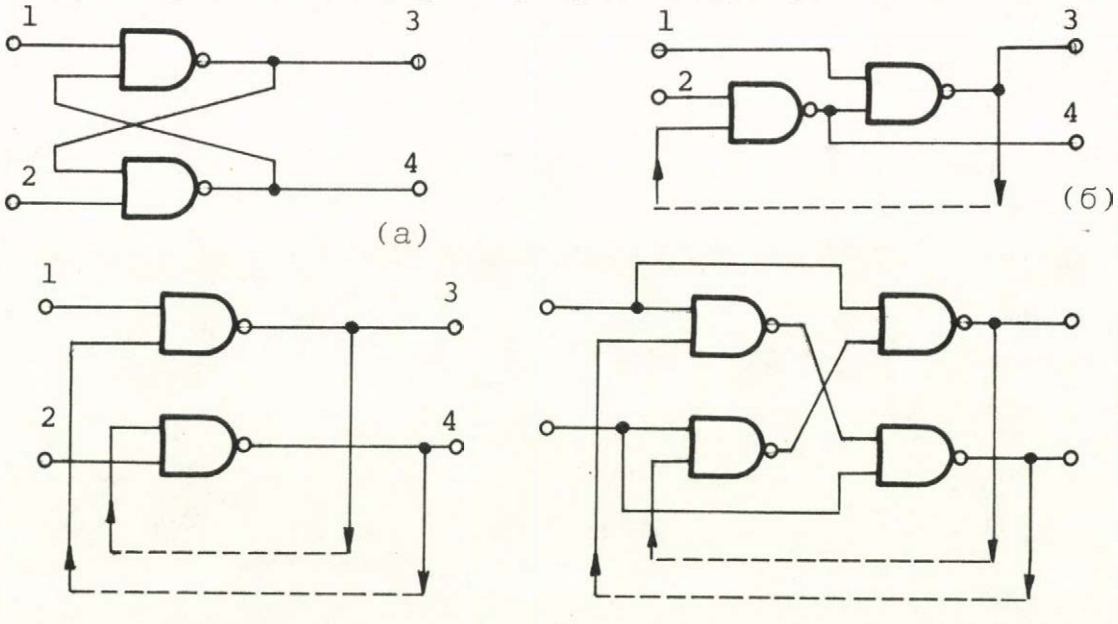
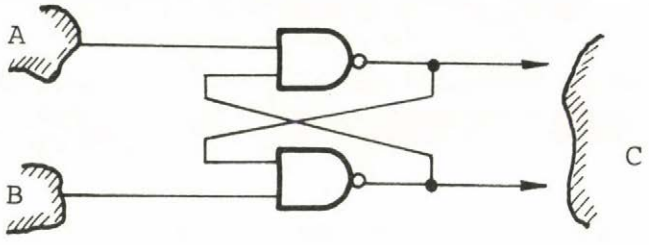


Рис. 2. Модели триггера RS.

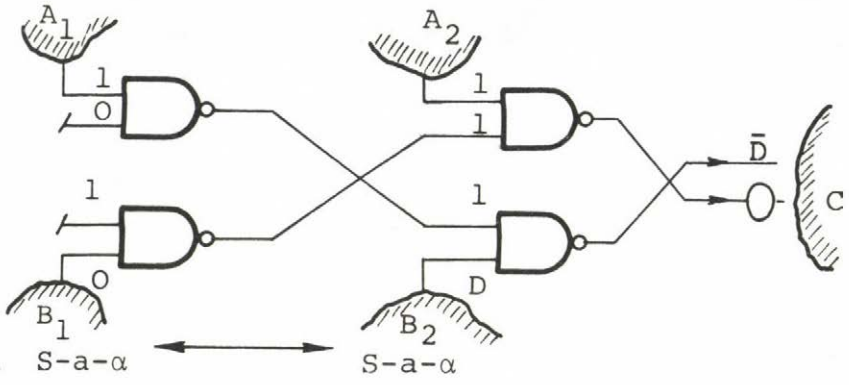
Итерационная модель асинхронной схемы должна, таким образом, выполнять ряд условий, вытекающих из сущности действия ее физического эквивалента.

1. Неисправность должна быть отображена в каждом элементе итерационной модели.
2. Во время траверсирования дорожек схемы значение D может в определенных условиях распространяться линиями обратных связей.
3. Заданное количество повторений должно удовлетворить требованиям стабилизации схемы /необходимо также принять во внимание опасность рисков/.

Модель, в которой не будет выполнено какое-нибудь из вышеуказанных требований может давать неправильные результаты. Рисунок 3



(а)



(б)

Рис. 3. Неправильная модель (б) триггера (а) , A, B, C - схема, A_i, B_i - элементы итерационной модели

иллюстрирует ненадежность модели Путзолу и Рота. В модели приведенные условия вызывают распространение значения D на наблюдаемый выход. Однако, заданное состояние является неправильным с точки зрения действия триггера, не выполняет условия стабилизации после включения линии обратной связи. Таким образом, тест является фальшивым. Вышеуказанная ошибка не появится при определении тестов для модели, показанной на рис. 2д. Эта модель является правильной.

Сейчас займемся второй из обсуждаемых проблем.

3. ТРЕХЗНАЧНАЯ МОДЕЛЬ

Будем описывать действие элементарных логических элементов в алфавите $V = \{0,1,v\}$, где 0 и 1 это логические значения, зато v обозначает неисправность. Интерпретация значения v следующая. Если в действующей правильно схеме сигнал принимает значение 0(1), а в неисправной - значение 1(0), то в трехзначной модели пары этих схем сигнала принимают значение v.

x_1	x_2	AND	OR	NAND	NOR
0	0	0	0	1	1
0	1	0	1	1	0
0	v	0	v	1	v
1	0	0	1	1	0
1	1	1	1	0	0
1	v	v	1	v	0
v	0	0	v	1	v
v	1	v	1	v	0
v	v	v	v	v	v

Рис. 4. Принципы действия вентилей, описанные в алфавите V.

На рис. 4 приведены таблицы истинности, определяющие действие логических элементов **AND, OR, NAND, NOR** в алфавите $\mathbb{V}$. На рисунке 5 же представлены примеры распространения неисправности $\mathbb{V}$ согласно вышеуказанным правилам. Анализ принципов распространения неисправностей приводит к следующим выводам:

Вывод 1 [3].

Тесты неисправностей в комбинационных схемах, свободных от разветвлений, определяемые с помощью вышеуказанной модели являются правильными.

Вышеупомянутый вывод проиллюстрирован на рис. 5а и 5в. Первый рисунок представляет случай, когда определенный тест касается неисправности одного типа, а второй - неисправности противоположного типа. Легко заметить, что определенный тест относится всегда к такой сложной неисправности, которую можно локализировать. Случай одиночных неисправностей является здесь тривиальным.

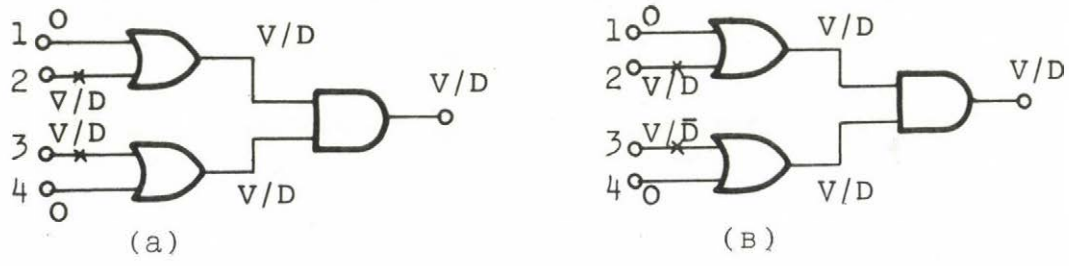


Рис. 5. Иллюстрация выводов 1;2

Вывод 2 [3].

Тесты неисправностей в комбинационных схемах с интерференцирующими разветвлениями, определяемые при помощи вышеуказанной модели правильны тогда, когда схема имеет так называемые "симметрические пути".

Вышеупомянутый вывод проиллюстрирован рисунком 5б. Симметрия путей состоит в том, что каждый из смыкающихся путей имеет идентичное число инверсий. Следовательно, неисправность линии разветвления может одновременно распространяться по нескольким путям без опасности колизии, которая показана на рис. 5д. В действительности неисправность **s-a-0** линии 2 доходит до вентиля **AND** как одновременная неисправность **s-a-0, s-a-1** его выходов. Это вызывает постоянный 0 на выходе логического элемента. Обсуждаемая модель слишком узка и не учитывает вышеуказанной возможности, приводя к неправильным результатам.

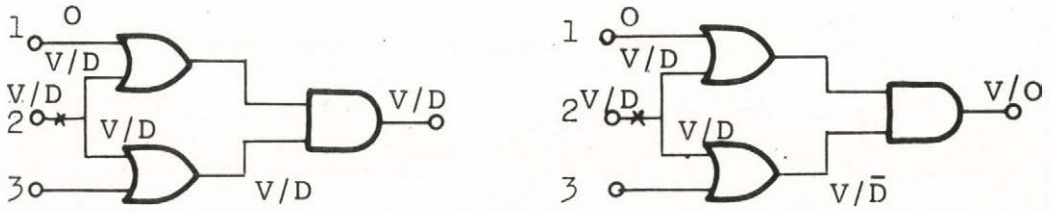


Рис. 5. Иллюстрация выводов 1:2

В заключении следует подчеркнуть, что приведенная трехзначная модель комбинационной схемы аналогична модели, полученной Эйхелбергдтом [4] и используемой для исследования рисков. Это позволяет соединить процесс моделирования схем с процессом распространения неисправностей для обсуждаемого класса комбинационных сетей.

К сожалению трехзначная модель ненадежная в случаях, представленных на рис. 5д.

4. ЧЕТЫРЕХЗНАЧНАЯ МОДЕЛЬ РОТА

В предыдущей главе указывалась необходимость расширения алфавита **V**, ибо введенная трехзначная модель оказалась ненадежной в случае схем содержащих разветвления и имеющих несимметрические пути.

Это расширение хорошо известно под названием алгебры Рота [2]. По этому поводу оно не будет обсуждаться в настоящей работе. Напомним только, что алфавит V расширяется до алфавита $V^* = \{0,1,D,\bar{D}\}$, а правила действия логических элементов. Очередной рисунок дает пример распространения неисправностей в вышеуказанной модели.

Основной недостаток алгебры Рота заключается в том, что если от источников неисправности до наблюдаемого выхода схемы ведет "n" различных путей, то число независимых испытаний перераспространения неисправности равно $\ell = \sum_{k=1}^{n} \binom{n}{k}$.

Это вытекает из необходимости составления одиночных возбужденных путей схемы, с целью получения возбужденных подграфов, так как существуют такие неисправности, которые могут распространяться только по избранному подграфу схемы.

Исследование всех возможных подграфов требует проведения ℓ испытаний перераспространения неисправностеи.

Вышеуказанный недостаток можно устранить расширяя описание тестируемой схемы.

5. РАСШИРЕННАЯ МОДЕЛЬ

В введенной нами модели действие схемы описывается при помощи алфавита $A \phi 2^{V^*}$. Это обозначает, что элементами алфавита A являются подмножества алфавита V^*.

На рис. 6 представлен алфавит A, а также расширенная таблица истинности для логического элемента NOR. Из этой таблицы вытекает, что если, например, одни из выходов вентиля NOR находится в состоянии 0^+ /что обозначает или 0 или D/, а другой в состоянии 0^- /что обозначает или 0 или $\bar{D}$/, то выход вентиля находится в состоянии X /что обозначает 0,1,D, или $\bar{D}$/. Функции F^u являются функциями определенными Ротом в D-алгебре. На этом основании получаем алгебраическое описание предыдущего примера ($G^{NOR}(0^+, 0^-) = X$).

$A = \{X,0,1,D,\bar{D},0^-,0^+,1^-,1^+,\varepsilon\} \subset 2^{V^*}$

$X = \{0,1,D,\bar{D}\}$

$D = \{D\}$

$\bar{D} = \{\bar{D}\}$

$0 = \{0\}$

$1 = \{1\}$

$0^+ = \{0,D\}$

$0^- = \{0,\bar{D}\}$

$1^+ = \{1,D\}$

$1^- = \{1,\bar{D}\}$

$\varepsilon = \{\ \}$

$G^u : A^n \to A,$

G^{NOR}	0	1	D	$\bar{D}$	0^+	0^-	1^+	1^-	X
0	1	0	$\bar{D}$	D	1^-	1^+	0^-	0^+	X
1	0	0	0	0	0	0	0	0	0
D	$\bar{D}$	0	$\bar{D}$	0	$\bar{D}$	0^-	1^-	0^-	0^-
$\bar{D}$	D	0	0	D	0^+	D	0	0^+	0^+
0^+	1^-	0	$\bar{D}$	0^+	1^-	X	0^-	0^+	X
0^-	1^+	0	1^-	D	X	X	0^+	0^+	X
1^+	0^-	0	0^-	0	0^-	0^-	0^-	0	0^-
1^-	0^+	0	0	0^+	0^+	0^+	0	1^+	0^+
X	X	0	0^-	0^+	X	X	0^-	0^+	X

1. $G^u(a_1,\ldots,\varepsilon,\ldots,a_n) = \varepsilon,$

2. $G^u(a_1,\ldots,a_n) = \{F^u(b_1^{j1},\ldots, b_n^{jn})\},$

$a_i \in A,\ b_i^{ji} \in a_i,\ i = 1,\ldots, n.$

$G^{NOR}(0^+,0) = \{F^{NOR}(0,0)\ ,\ F^{NOR}(0,\bar{D}),\ F^{NOR}(D,0)\ ,$

$F^{NOR}(D,\bar{D})\} = \{1,D,\bar{D},0\} = X$

Рис. 6. Расширенный алфавит.

Введение расширенного алфавита A ведет к существенному упрощению процесса определения тестов. Можно доказать, что применение определенной конструкции для получения возбужденных подграфов ведет в случае алфавита A исключительно к возбужденным путям. Применение алфавита A делает возможным распространение неисправностей более сложными подграфами. В общем, применение расширенной модели дает эффект одновременного возбуждения многих путей /подграфа/, что, в свою очередь, обеспечивает редукцию числа независимых испытаний распространения неисправности с ℓ до n.

При больших схемах и кратных неисправностях /что имеет место для асинхронных автоматов/ эта редукция очень существенна.

6. РЕЗЮМЕ

В Институте Вычислительной Техники Варшавского Политехнического Университета разработана система программ, определяющая тесты для неисправностей типа "постоянное логическое значение" в асинхронных автоматах. На рис. 7 приведены ограничения системы. В системе образуется интерационная модель автомата, а также применяется расширенный алфавит A. Это решение представляется нам наиболее правильным.

I - число внешних входов,
O - число внешних выходов,
F - число линии обратной связи,
G - число элементов,
P - число итерации,

1. $I \leq 100, \quad O \leq 100,$
2. $G \leq 1000,$
3. $G \cdot P \leq 1000,$
4. $(F+O) \cdot G \leq 4000.$

Рис. 7. Ограничения системы.

Если бы принять, что разработка системы состоит из трех этапов:

1. концепции,
2. осуществления,
3. проверки истинности,

то первых два этапа мы уже закончили, а третий значительно продвинут вперед.

[1] Putzolu G.R., Roth J.P.: A heuristic algorithm for the testing of asynchronous circuits, IEEE Trans. Comp., Vol. C-20, June 1971.

[2] Roth J.P.: Diagnosis of automata failures: a calculus and a method, IBM Journal, July 1966.

[3] Sapiecha K.: Zastosowanie wyróżnika sieci kombinacyjnuch do wyznaczania jej testów diagnostyczanych, AAiT z.3, 1976.

[4] Eichelberger E.B.: Hazard detection in combinational and sequential switching circuits, IBM Journal, March 1965.

[5] Breuer M.A.: The effects of races, delays and delay faults on test generation, IEEE Trans. Comp. Vol. C-23, Oct. 1974.

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1977

АЛЕКСАНДР АЛЬБИЦКИ КРИСТОФ ЯСИНСКИ

ПРОЕКТИРОВАНИЕ САМОСИНХРОННЫХ СХЕМ

Институт Телекоммуникации Варшавской Политехники

I. ВВЕДЕНИЕ

Реализация автомата без особого кодирования входных переменных - например без указания тактовой переменной - приводит к асинхронной схеме. Схемы эти имеют тот недостаток, что кодирование внутренних состояний становится громозким, так как надо устранять опасные состязания. В настоящей статье представлен будет метод реализаций асинхронного автомата без обязательности устранения опасных состязаний, так что кодирование становится значительно проще. Метод обоснован на перемене асинхронного способа действия в самосинхронное, что выполняется предварительно закладывая логическую структуру схемы. Основное действие схемы состоит в том, что при каждой перемене на входах схемы генерируется импульс, который употребляется как тактовой импульс. Мы представим структуру схемы, анализируем её работу, поставим алгоритм проектирования, и сделаем пример реализации асинхронного автомата работающего как самосинхронный. В заключении мы постараемся доказать, что использование в практике самосинхронных схем не только возможно, но и даёт хорошие результаты.

2. АНАЛИЗ РАБОТЫ САМОСИНХРОННОЙ СХЕМЫ

Принимаем, что реализовать будем автомат заданный нормальной, минимальной таблицей переходов. Нормальная таблица это такая, в которой каждый переход ведёт непосредственно в стабильное состояние, и при этом на выходе может произойти не больше одной перемены.

Самосинхронная схема представлена на рис. I.

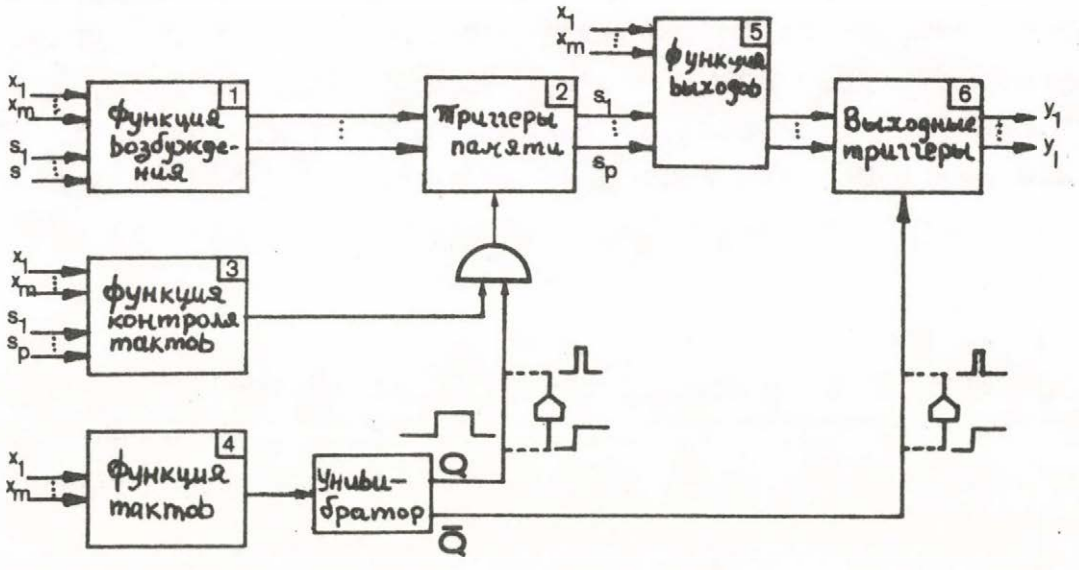


Рис.I. Самосинхронная схема.

Схема действует следующим способом:
- Блок 4 вместе с унивибратором генерирует при каждой перемене входных переменных, короткий импульс который мы называть будем тактовым импульсом. Если на входах появится „одновременно" несколько перемен, то генерирован будет лишь один тактовой импульс (свойство унивиоратора). Такой режим работы характерен для синхронных схем. В асинхронных схемах одновременная перемена на нескольких входах не допускается, так как приводит это к помехам в действии схемы. Генерирование тактового импульса и управление при его помощи работой Блока 2 - блока триггеров памяти - позволяет минимизировать таблицу переходов. Минимизация ооснована на том, что если в таблице переходов существует такое стаоильное состояние, которое является стабильным только в одном столбце, то в нормальном режиме работы это состояние недостижимо. Оно недостижимо, так как длительность тактового импульса всегда короче времени перехода из одного внутреннего состояния в

другое. Можно затем символ стабильного состояния заменить вимволом „—". Благодаря этому получается возможность добавочной минимизации.

- Тактовые импульсы блокированы (схема И) тогда, когда входные переменные не переводят автомата из одного состояния в другое. Это даёт возможность упрощения функции возбуждений. Все символы стабильных состояний можно теперь заменить символом „—", так как в стабильных состояниях триггеры памяти не действуют. Очевидно надо также реализировать функцию управления тактовыми импульсами.

Блок 3. Способ определения этой функции подан будет дальше в примере. Надо заметить, что не рекомендуется в общем случае заменять сразу все символы стабильных состояний символом „—", так как тогда получается более сложная функция управляющая тактовым импульсом.

- В блоке 2 могут быть использованы синхронные триггеры или же регистры сдвига (с выходами с каждой ячейки).

- Для правильного функционирования схемы на её выходах находится Блок 6 - выходных триггеров. Триггеры эти синхронизированы отрицательным склоном тактового импульса, а их задание состоит в ликвидировании риска в выходных функциях. Для обеспечения правильного действия самосинхронной схемы представленной на рис.I надо выполнить следующие условия:

$$t_{3\,макс} + d \leq t_{4\,мин} + \Delta + \sigma \qquad (1)$$

$$t_{1\,макс} + d + \varepsilon \leq t_{4\,мин} + \Delta + \sigma + t_{u\,мин} \qquad (2)$$

$$D \geq t_{u\,макс} + \varepsilon + в_{макс} + t_{5\,макс} \qquad (3)$$

где: d - это расстояние на шкале времени между первой и последней переменой при „одновременных" переменах на входах схемы,

D - длительность тактового импульса,

σ - время перехода импульса через унивиоратор,

ε - максимальное время включения триггеров,

$t_{u\,мин\,(макс)}$ - минимальное (максимальное) время перехода через схему И,

$t_{k\,мин(макс)}$ - минимальное (максимальное) время перехода через к-тый блок,

- искусственное запаздывание употребляется только тогда, когда необходимо,

$в_{макс}$ - максимальное время переключения триггеров.

Теперь можно пояснить значение поданных выше условий. В условии (1) требуется, чтобы дорога (в смысле времени перехода этой дороги) через блок 3 была короче дороги через блок 4, блок запаздывания Δ (в практике можно почти всегда принимать $\Delta = 0$) и унивибратор. Условие (2) требует, чтобы дорога через блок 1 была короче дороги через блок 4, блок запаздывания Δ, унивибратор и схему И.
Третье условие обозначает, что длительность тактового импульса больше чем время перехода дороги через схему И, блок 2 и блок 5.
Можно также оценить быстродействие схемы, оценивая минимальное время необходимое для выполнения одного действия. Это время (T) задаётся следующей неравностью:

$$T \geq (t_{4\,макс} + \Delta + \sigma + D + \varepsilon) - t_{5\,мин}$$

По правой стороне неравенства находится разница времени перехода дороги через блок 4, блок запаздывания, унивибратор (вместе с длительностью тактового импульса и дороги через блок 5). Условие это обозначает, что импульс проходящий нижней дорогой подаётся на вход синхронизации блока 6, когда поданы уже на входы этого блока сигналы приходящие из блока 5.

3. МЕТОД ПРОЕКТИРОВАНИЯ САМОСИНХРОННЫХ СХЕМ.

Проектирование самосинхронных схем может быть выполнено на основе следующего алгоритма:

I. определить в минимальной таблице переходов все те стабильные состояния, которые являются стабильными только в одном столбце. Вместо этих состояний поставить " —— ";

2. минимизировать таблицу переходов;
3. определить таблицу функции управляющей тактовым импульсом. Таблица эта имеет размеры таблицы переходов, а восстанавливается ее следующим способом: в места соответствующие стабильным состояниям таблицы переходов ставится 0, в места соответствующие нестаоильным состояниям ставится I, места с „—" остаются без перемен;
4. на основе полученной в 3 таблице определить функцию управляющую тактовым импульсом;
5. в таблице переходов в место стаоильных состояний поставить „—";
6. следующие этапы синтеза исполнять так, как для синхронных схем.

4. ПРИМЕР

Пусть задана будет минимальная таолица переходов-выходов асинхронного автомата (таолица I).

Таолица I

S \ x_1x_2	00	01	11	10
1	①,00	2	–	①,11
2	3	②,—	–	1
3	③,11	5	–	–
4	④,–0	④,00	–	1
5	4	⑤,11	–	–

Стабильные состояния ② , ③ и ⑤ заменить можно символом „—" и минимизировать таблицу. В результате получаем таблицу 2.

Таблица 2

	x_1x_2 / s	00	01	11	10
	1	(1),00	2	–	(1),11
2,3 →	2	(2),11	5	–	1
	4	(4),–0	(4),00	–	1
	5	4	(5),11	–	–

На основе таблицы 2 определить можно функцию управляющую тактовым импульсом C . Для этого надо закодировать внутренние состояния (кодирование может быть произвольное - в нашем случае выбираем I ⟶ 00, 2 ⟶ II, 4 ⟶ 01, 5 ⟶ 10), построить таблицу функции C (таблица 3) и определить выражение описывающее эту функцию (4).

Таблица 3

y_1y_2 / x_1x_2	00	01	11	10
00	0	1	–	0
01	0	0	–	1
11	0	1	–	1
10	1	0	–	–

$$C = x_2\bar{y}_1\bar{y}_2 + x_2y_1y_2 + x_1y_2 + \bar{x}_2y_1\bar{y}_2 \qquad (4)$$

Теперь вместо всех стабильных состояний в таблице 2 вставить можно „—" (таблица 4) и определить функции возбуждения триггеров памяти (5) и (6) (выбираем триггеры типа D) и функции выходов (7).

Таблица 4

y_1y_2 / x_1x_2	00	01	11	10	00	01	11	10
00	–	11	–	–	00	–	–	11
01	–	–	–	00	–0	00	–	–
11	–	10	–	00	11	–	–	11
10	01	–	–	–	–	11	–	–
	Y_1Y_2				z_1z_2			

$$Y_1 = x_2 \qquad (5)$$

$$Y_2 = \overline{y}_2 \qquad (6)$$

$$z_1 = z_2 = y_1 \qquad (7)$$

При определении этих функций не надо заниматься влиянием риска на действие схемы, так как риск устраняется за счёт синхронной работы всей схемы.

5. ЗАКЛЮЧЕНИЕ

Использование самосинхронных схем зависит от нескольких условий. Обсудим два вопроса: вопрос простоты проектирования и вопрос технической реализации самосинхронных схем. Проектирование в нашем случае становится чисто логическим (без особых рассчётов временных условий - так как они выполнены конструкцией схемы) и сводится к минимизации числа внутренних состояний, к простому кодированию (так как отсутствуют состязания) и к определению функций возбуждений и функции управляющей тактовым импульсом. Все эти задачи алгоритмичны и легко выполняются на ЦВМ. Затем проектирование самосинхронных схем является значительно проще чем проектирование асинхронных автоматов.

Техническая реализация самосинхронных схем, судя по блочной схеме представленной на рис.I, довольно сложна. Но мы представим ниже на рис.2 схему генерирующую тактовый импульс и если такие схемы изготовлены были бы целиком в стандартных технологических пакетах, то конструкция схемы стала бы совсем простой и сводилась бы до реализации блоков возбуждения и функции C, а это, как видно по примеру, несложно.

Заметим, что в технической реализации блоков памяти и выходов используются синхронные триггеры.

(Если используются триггеры типа МАСТЕР-СЛЕЙВ, то необходимо за унивибратором употребить дифференциальные схемы и тогда выходами будут I' и 2').

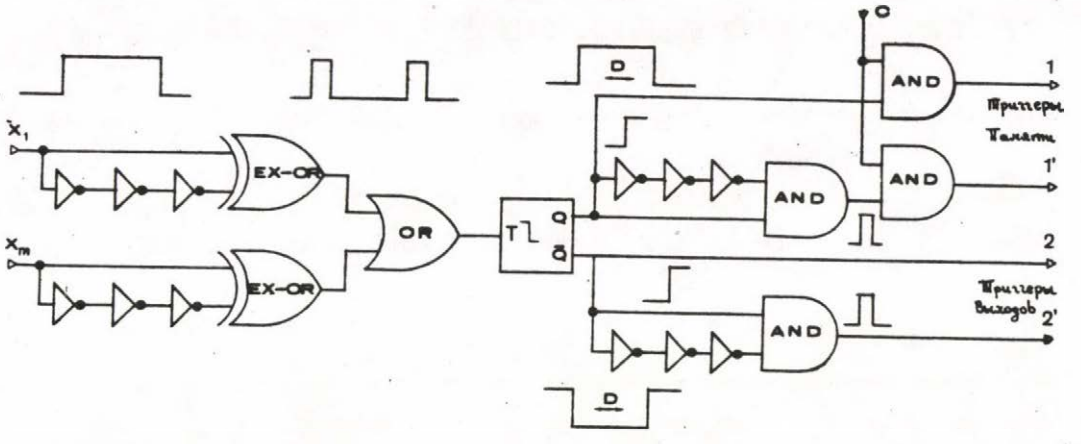


Рис.2. Техническая реализация генератора тактовых импульсов

Это является тоже преимуществом самосинхронных схем реализуемых на интегральных схемах типа ТТЛ. Так как кодирование внутренних состояний произвольно, то можно в блоке памяти применять сдвиговые регистры с выходами из каждой ячейки.

Сравнивая асинхронные и самосинхронные схемы по быстроте действия заметим, что для сложных вариантов кодирования (много переходов через нестабильные состояния) асинхронных схем их быстрота действия сравнима с быстротой самосинхронных схем. При реализации самосинхронных схем на интегральных схемах ТТЛ получается максимальную быстроту действия до ок. 10 мгц.

Принимая во внимание все выше указанные условия, мы надеемся, что самосинхронный режим работы автоматов и связанный с этим представленный нами метод проектирования использо-

ваны оудут в практике.

1. W. Majewski; A. Albicki: "Projektowanie telekomunikacyjnych układów cyfrowych" Wkił Warszawa 1977.

2. A. Albicki; K. Jasiński: "On quasi-synchronized sequential circuits". 1976 European Conference on Circuit Theory and Design Genova.

3. H. Chuang, S. Das: "Synthesis of multiple-input change asynchronous machines using controlled excitation and flip-flops", IEEE Trans. Comput. vol C-22 pp.1103-1109, Dec. 1973.

4. Ch. Rey; J. Vaucher: "Self-synchronized asynchronous sequential machines" IEEE Trans.Comput. vol. C-23 pp.1306-1311 Dec.1974.

5. J. Brederson: "Comments on "Synthesis of multiple-input change asynchronous machines using controlled excitation and flip-flops and "authors´reply" IEEE Trans.Comput. vol C-24 Nov.1975.

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1977

МАРЕК ПЕРКОВСКИ

СИСТЕМА АВТОМАТИЧЕСКОГО ПРОЕКТИРОВАНИЯ ЦИФРОВЫХ УСТРОЙСТВ

Институт Автоматики
Варшавского Политехнического Института

ВВЕДЕНИЕ

В настоящей работе обсуждаются основные черты системы автоматического проектирования цифровых устройств **DIADES.** Система применена в Институте Автоматизации Варшавского Политехнического Института на ЭВМ **ODRA 1325** и **CDC CYBER 73** на языках **LISP 1.5.9, FORTRAN 1900** и **PLAN.** Задача системы - сделать возможным синтез не очень сложных устройств из интегральных модулей **TTL** малого и среднего уровня интеграции на основе описания действия этих устройств на языке **ADL.** Подробное описание системы, применяемых в ней языков, теоретических основ, методов проектирования и полученных результатов можно найти в цитированной литературе.

1. ОПИСАНИЕ ЗАДАЧИ. ЯЗЫК ADL

Системы автоматического проектирования должны употреблять, для описания проектируемого цифрового устройства, специальные языки описаний, в том числе - язык межрегистровых посылок.

Язык **ADL** специально разработан для потребностей нашей системы [20, 21, 12, 3, 4] . Он по сравнению с известными языками этого типа является языком имеющим относительно развернутую грамматику, большую эластичность и разнообразие средств описания. Автомат может быть описан на алгоритмическом, микропрограммном и структурном уровнях. Язык имеет богатые средства для описания параллельных процессов и временных процессов, арифметических и логических действий, а также для описания структур устройств. Транслятор языка **ADL** обеспечивает богатую диагностику

ошибок и тоже автоматическую коррекцию некоторых синтактических ошибок.

Программа на языке **ADL**, это сформализированное описание граф-схемы алгоритма с параллельными ветвями, описывающей поведение проектируемого устройства. Это описание определяет также в какой-то степени структуру устройства. Подробность этого описания может быть частично корректирована проектировщиком. Например, он может описывать Булевы функции на словах или на единичных разрядах.

Принято, что программы на языке **ADL** могут быть написаны проектировщиком незнающим технологии проектированных в промышленности блоков ни специфических алгоритмов синтеза. Он должен только знать синтактику и семантику языка **ADL** и уметь выразить на нем поведение проектируемого устройства. С другой стороны, проектировщик знаком с цифровой техникой, типичными блоками и устройствами, а также с особенными приметами компилятора языка **ADL** и других программ системы, может написать программу поведения, которая будет лучше имплементированной системой.

Значение системы на языке **ADL** другое чем в аналогических языках в известных автору системах автоматического проектирования, в которых программа на входном языке точно, "жестко" описывает структуру операционного устройства и контролирующего автомата, оставляя синтезирующим программам системы возможность только уточнения этой структуры, а не ее трансформации.

Программа на языке **ADL** транслируется в описание на язык внутримашинных соотносительных структур **GRAF**, которое является основой трех главных программ системы: программы структурного синтеза управляющего автомата, программы имплементации /внедрения/ операционного автомата, программы трансформации и оптимизации.

Абстрактивная имплементация совершается программой **IMPLEM**

[35, 3] . Она осуществляет перевод инструкций графсхемы с языка GRAF в описание структуры, состоящей из абстрактивной блоков / под абстрактными блоками разумеем здесь блоки с памятью, или комбинационные, такие как: счетчики, сумматоры, мультиплексеры, логические вентили - если для них не специфирована внутренняя структура и число входов и выходов/. Эта структура записывается в виде графа. Вершины этого графа соответствуют абстрактивным блокам; стрелки - соединениям между блоками. Из выходов абстративной блоков может отбираться цифровая информация, являющаяся какой-то функцией информации подаваемых на их входы.

В графе сети действий, создаваемом на основании программы в ADL существуют два типа узлов - синхронные и асинхронные. Самовыполнение синхронных узлов требует не менее одного такта основных часов. Некоторые команды имплементируются как синхронные, другие как асинхронные причем проектировщик может декларировать асинхронное выполнение команд обычно имплементированных как синхронные при употреблении оператора ASY. Непосредственная первоначальная имплементация программы, в которой имеются сложные команды посылок и условные команды дает скоростную систему, но с увеличенным количеством оборудования, потому, что каждая команда осуществляется во время одного такта. Очевидные трансформации и имплементации ведут к получению все более медленных и дешевых вариантов, вплоть до получения варианта квазиоптимального при предположенной критерии. Преобразования разбивают однотактные команды на ряд однотактных команд и заменяют команды другими, меняя одновременно структуру автомата, выполняющего программу. Изменение структуры может быть кроме того результатом различных имплементаций той же команды.

Следовательно в отличии от других языков межрегистровых посылок программа на языке ADL не определяет твердым образом временные отношения и структуру устройства, а только описывает его поведение, то есть логическую и временную зависимость входных и выходных переменных.

Таблица № 1

№№	Название программы	Время выполнения в сек.	Цена выполнения программы в SS [1]	Число простых команд	Число сложных команд	Число узлов в графе	Число узлов в абстрактной имплементации
1	MNUL 1	16.7	65.0	14	4	7	7
2	MNUL 2	17.1	67.1	11	4	7	7
3	CODDEC	16.2	63.9	15	7	9	8
4	DECCOD	30.0	119.0	16	5	17	23
5	KLAS 1	19.3	76.2	12	5	15	8
6	KLAS 2	18.7	74.0	14	2	10	8
7	KLAS 3 [2]	19.7	77.8	13	7	13	12
8	OPIMP [2]	-	-	14	8	13	10
9	TICKET [2]	-	-	37	12	30	26
10	MICR 1	49.6	197.1	38	10	27	35
11	MICR 2	38.4	152.2	36	10	25	27
12	MICR 3 [2]	31.0	122.8	17	7	19	21
13	SWIATLA [2]	92.2	366.5	72	40	78	92
14	FILTR [3]	18.0	70.7	8	1	-	24
15	ADDER 1 [2][3]	45.0	178.7	7	5	-	9
16	ADDER 2 [2][3]	43.0	171.4	8	6	-	9

1) SS – системные секунды, в число которых входит время выполнения программы и оплата за занятие памяти и устройств ввода и вывода

2) – программа содержит подпрограммы

3) – программа описывающая структуру цифрового устройства

Время компиляции и абстрактивной имплементации является относительно небольшим и линейно растет с длиной программы. Зато времена оптимизирующих и имплементирующих программ сильно зависят от специфики задачи и обычно растут показательно вместе с ростом длины входной программы.

В таблице 1 составлено некоторые сравнительные данные касающиеся некоторых прогамм на языке **ADL. MNUL** является описанием устройства для параллелного умножения чисел в бинарном коде. **CODDEC** является описанием секвенциального декодирующего устройства превращающего числовое значение записанное в коде **BCN** в равное по величине числовое значение в коде **BCD. DECCOD** выполняет операцию противоположную **CODDEC. KLAS** является описанием автомата для классификации резисторов касательно отклонения их сопротивления от нарицательной величины [15, 17, 7, 20] . **TICKET** является описанием автомата который высчитывает значение сдачи и выдает ее в соответственных монетах вместе с железнодорожным билетом [19, 20] . **MICR1, MICR2, MICR3** это три описания того же компьютера на различном уровне детализации [21]. **SWIATLA** является описанием управляемой компьютерной цифровой приделки для устройства координации системы регулирования уличного движения [12] (в цитируемой литературе обсуждается оптимизация этих систем и/или представлены описания на языке **ADL** - кроме того все примеры подробно обсуждены в [3] и [25]).

2. ОПТИМИЗИРУЮЩИЕ ТРАНСФОРМАЦИИ

Существенной чертой системы, отличающей ее коренным образом от известных систем этого типа, является возможность автоматических трансформаций и оптимизаций автоматов на уровне программ их действия. Программы оптимизации опираются на эвристический поиск в пространстве эквивалентных решений и используют среди других: операторы согласовывания с эталоном, операторы всегда применяемые, распознование сходства структур, действия на описывающих программы формальных соотношениях, а также автоматическое доказательство теорем /метод резолюции Робинсона/ [1, 6, 7, 15, 16, 17, 19, 20, 23, 39] .

Трансформации применяемые в системе **DIADES** имеют два типа: "вегда применяемые" /это например: упрощающие трансформации исключающие избыточность или нормализующие данные/, а также трансформации реализующие специфические цели. К трансформациям всегда применяемым принадлежат трансформации, упрощающие буле - вых и арифметических выражений, нормализация описаний команд, переход от сети действий с параллельными ветвями к таблице переходов автомата , трансформации комбинационных схем предшествующие имплементацию интегральными модулями /имплементация эта заключается в переходе от описания структуры из абстрактных блоков к структуре из интегральных модулей **TTL** малого и среднего уровня/.

Трансформации реализирующие цели разделены на группы отвечающие этим целям. Большинство из этих трансформаций сводит к минимуму числа элементов, стоимости системы или ее скорости. Примерными трансформациями этого типа являются: переход от автомата Мура к автомату Мили, подстановки в арифметических выражениях, преобразования заменяющие память логикой , введение петли и переменных индексированных соединения общих подграфов, минимизации графа соотношений для автомата.

Трансформации можно разделить учитывая способ их осуществления, еще по-другому: трансформации надписей а также трансформации соотносительных структур /особенно графов/ [16] . Первая группа функционирует на описаниях команд, описаниях узлов структур, условных выражениях и декларациях программ; вторая на структурах и графах. Среди трансформаций выделяются трансформации локальные и валовые. К первой группе принадлежат, например, преобразование согласно эталону $p \wedge \neg p = \emptyset$; ко второй оператор **WALKER**, который разыскивает подобные подграфы программы, организует программные петли и вводит индексированные переменные [7, 19] . Специфическими типами трансформации являются трансформации доказывания правильности программ и параллельных программ управляющих автоматов, а между ними трансформации вычисляющие исходные соотношения и входно-выходные соотношения.

Программа на языке **GRAF** трансформируется для доказывания ее правильности, для одновременной оптимализации, а также для обнаруживания временных конфликтов между переменными, описывающими входные и внутренние сигналы автомата [23, 27] .

Эти трансформации основаны на теории семантики программ развитой Мазуркевичем и Бликлем. Расширение применения этой теории на параллельные программы реального времени описано в [23, 27].

Оптимизирующие программы используют стратегию эвристического поиска в пространстве эквивалентных решений /программа, структур/. Они ищут решения, которые минимизируют поставленную целевую функцию [17, 7, 1, 30] .

Оптимизация проводится тремя способами:

- при использовании рядов небольших "локальных" операторов подобранных для реализации определенных целей;
- при использовании больших "валовых"/"глобальных"/процедур, трансформирующих целые программы;
- при использовании программы автоматического доказательства теорем **CHANG**.

Потому что действие программы **CHANG** не является эффективным для больше чем 20 клаузул, получает она при каждом вызове только небольшие наборы клаузул соответствующих аксиомам, теоремам и гипотезе. Гипотеза и соответствующие другие входные данные для программы **CHANG** выбираются более существенной эвристической программой [5, 23] .

Из существующих до сих пор опытов этой системы вытекает, что наиболее критическим принимая во внимание время и память машины является действие оптимизирующих программ. Примеры действия этих программ указаны в [19, 1, 6, 5, 7, 23, 27] .

В одном из экспериментов описанных в [19, 20] , программа для машины **CYBER** 73 при стоимости примерно 720**SS** нашла 4 существен-

но разные варианты программы TICKET, для которой первоначалный граф имел 30 узлов /люди решали эту проблему около 1÷3 часов/. Хотя на основе литературы известно, что программа подобного типа /в которых выступает приспособление к эталонам/ являются относительно медленными, кажется, что возможно создание структур и функции, которые значительно ускорят действие оптимизирующих программ.

3. ЕДИНАЯ МЕТОДИКА СИНТЕЗА

Среди программ входящих в систему автоматического проектирования цифровых устройств можно выделить два основных типа - программы трансляции /в общих чертах - переходы из одного языка описания в другой/ и программы поиска решений. Большинство из этих последних можно привести к следующей общей модели:

- имеются правила для генерации определенного множества S в котором содержится множество всех решений задачи. Множество это можеть быть конечное или нет. В случае когда множество конечное не всегда можно определить его мощность без хотя бы частичной генерации решений;
- имеются условия ограничивающие, не выполнение которых вызывает отбрасывание множества $S' \subset S$ из множеста потенциальных решений;
- имеется функция оценок определяющая качество элементов монжества S.

Нужно найти решение, т.е. элемент множества S выполняющий ограничения, и которому отвечает минимальное значение функции оценок.

Для задач этого типа возможны три общих подхода

- разработка детальных алгоритмов для задач, учитывающих их специфические черты, но не учитывающих эвристические директивы;
- сведение к задачам теории графов, целочисленного или динамического программирования;

- разработка алгоритмов опирающихся на методы эвристического программирования, учитывающие общие и специфические эвристические директивы.

Первый подход учитывает специфику задач, однако чаще всего в логическом а не эвристическом смысле, является все таки малоэластичным на изменение функции цели или ограничения. Второй подход не учитывает специфики задач, в результате чего часто уменьшается его расчетная эффективность.

Автору кажется, что учитывая специфику систем автоматического проектирования цифровых устройств, очень подходящим и эффективным орудием для решения представленных задач являются методы эвристического программирования применяемые в искусственном интеллекте - а особенно методы поиска в пространстве состояний.

Эти методы делают возможным как нахождение квазиоптимальных так и оптимальных решений, а также широкое использование общих и специфических эвристик. Первые попытки такого подхода можно найти в работах Давидсона, Детмаера и Шнейдера, Слагля, Шмидта и Друффеля. Также в системе **DIADES** целый ряд алгоритмов опирается на поиске в пространстве состояний. Алгоритмы эти служат примерно для: генерации импликантов, разрешения проблем покрытия и покрытия с закрытием, проблемы окрашивания карт, проблем ведения соединений и размещения элементов, проблем декомпозиции и распределения, кодирования и минимализации автоматов, синтеза сетей **TANT**.

Сущность применяемого подхода объясняется на примере. При поисках покрытия булевой функции простыми импликантами о минимальной суммарной стоимости: множество **S** - это множество всех импликантов функции, целевая функция - это сумма стоимости импликантов с избранного покрытия **S'**, где $S' \subseteq S$. Процесс решения заключается в очередном выборе "лучших" соответственно некоторых правилам и эвристикам импликантов и с одновременных исключением других импликантов покрывающих меньшее количество минтермов функции /подчиненных импликантов на данном этапе/.

Таким образом создаются очередные состояния пространства состояний, и считаются для них целевые функции. После получения первого решения значение его целевой функции становится величиной ограничения. Наступает отсутствие решений в дереве и создание его новых поддеревьев. Доходя к состоянию о целевой функции не меньше чем прежде найденная, наступает отрезание поддерева начинающегося с этого состояния. После возврата к инициальному состоянию - последнее найденное решение является минимальным решением. Операторами пространства являются импликанты, состояниями подмножества инициального множества импликантов, а решениями - заключительные состояния ветви дерева, в которых не совершилось отрезание.

Эта модель является общей в случае синтеза цифровых устройств. В ситеме **DIADES** образована универсальная программа **MULTICOMP**, которая служит для развития пространства состояний при употреблении различных стратегий развития, функций цели и ограничений. Выбор стратегии /в том числе **first-breadth, depth-first, ordered-search, branch-and-bound**/ производится на основе множества параметров, которые могут быть динамически изменяемы во время процесса поиска по дереву. Среди методов наилучшей оказывается стратегия **branch-and-bound.** Преимущество этого метода заключается в том, что:

- можно в нем относительно нетрудным способом учесть в программе разные общие и специфические эвристические директивы, которые существенным образом ограничивают процесс поиска в пространстве решения задачи;
- относительно быстро получается решение квазиминимальное, затем поочередно все лучшие решения и в конце решение минимальное /следовательно здесь возможно получить по крайней мере часть решений, в то время как другие методы практически могут не дать решения вообще/;
- метод требует относительно небольшой емкости памяти машины в сравнении с другими методами гарантирующими минимальное решение;

- возможной является замена между степенью минимальности решения и скоростью его получения; это означает, что если декларируем, что результат может быть худшим $\alpha\%$ от минимального решения, то соответственно быстрее мы получим решения дающие эту гарантию;
- возможно применение разных функций цели, функции оценок и ограничений, изменение операторов и правил генерации частичных решений /квазирешений/ не меняя общей стратегии; эта эластичность метода очень важная во многих применениях.

Каждая задача синтеза для программ **MULTICOMP** изображается через: описание инициального состояния, операторов, ограничивающих условий, условий определяющих решение и функции испытывающих отношения между операторами. К этим последним принадлежат: второстепенность операторов, локальная и валовая эквивалентность операторов и независимость операторов.

Подробное описание концепции программы **MULTICOMP**, а также ее машиной имплементации, примеров применения и обслуживания полученных результатов можно найти в [28, 29, 30, 10].

По мнению автора **MULTICOMP** является интересным инструментом по следующим причинам:

а/ дает возможность единого подхода к широкому классу комбинаторных задач, что имеет с одной стороны теоретическое значение /например, для исследований сложности и классификации деревьев, разрешения задач различных классов, для доказывания теорем о возможностях ограничения деревьев для классов задач об определенной специфике/, как и практическое /сравнение эффективности разных методов разрешения того же, сравнение эффективности стратегии перед написанием специализированных программ, упрощение и унификацию процесса перехода от условий задачи к действующей программе/.

б/ делает возможным сравнение и перенесение методов разрешения из одного класса задач в другой, через нахождение общих методи-

ческих и эвристических директив.

в/ имеет дидактическое значение, так как систематизирует и разрешает в едином изложении разные вопросы , указывая на их общие черты и комбинаторный характер;

г/ для ряда задач представляется как эффективное орудие синтеза /например, минимализация автомата с 6-ю состояниями занимает, вместе с компиляцией, примерно 20 секунд на цифровой машине **ODRA 1325**/; для других его результаты являются полезными при создавании новых, специализированных алгоритмов.

4. СТРУКТУРНЫЙ СИНТЕЗ

Подсистема структурного синтеза автоматов реализуется на машине **ODRA 1325** на языках **FORTRAN** и **PLAN**. Система эта состоит из иерархии программ, из которых каждая реализует определенный уровень синтеза. Система позволяет на относительно нетрудное программирование новых алгоритмов благодаря эластичной, модульной структуре. Самый низкий уровень иерархии составляют программы в **PLAN**. Программы эти реализуют операции на отдельных машинных битах и словах, что делает возможной эффективную конструкцию данных и алгоритмов для логического синтеза. В состав этого уровня входят программы совершающие логические действия на словах и таблицах, простого и циклического перемещения, оборачивания слов нулево-единичных, действий на нулево-единичных шнурах /включение, исключение и выбор определенных шнуров/, и т.д. Второй уровень иерархии составляют процедуры на языке **FORTRAN** реализующие операции на так называемых кубиках и таблицах кубиков. Третий уровень составляют процедуры на языке **FORTRAN** реализующие ряд известных алгоритмов синтеза комбинационных структур /Квайна-Мак Класки , Зиссоса, Казакова и т.д./, а также новые алгоритмы синтеза. Процедуры второго и третьего уровней заключают в себе соответственно модифицированную систему процедур, Рота, Детмаера и Су с тем, что применено к ней бинарное кодирование опирающееся на теорию Улуга. Аналогическим способом запрограммированы процедуры для трехзначной логики. В состав второго уровня входят процедуры чтения и пе-

чатания таблиц кубиков; определения, копирования, суммирования, произведения, запрета, поглощения, разрешения, абсорбации и переписывания таблиц; вычисления значения некоторых параметров определяемых таблицами, трансформации таблиц /замена, исключение и включение колонн/ и др. Подробное описание процедур второго и третьего уровня можно найти в работе [14].

Подсистема структурного синтеза имеет два входа: таблиц переходов/выходов автоматов и логических функций в виде матриц гиперкубов. В состав подсистемы входят следующие программы: перехода от граф-схемы алгоритма к автомату /Мура, Мили/, трансформации автомат Мура автомат Мили, минимизации не вполне определенного автомата, кодирования автомата, вычисления функции возбуждений, синтеза комбинационных схем. Ряд известных алгоритмов синтеза комбинационных схем запрограммирован опираясь на набор программ из системы Детмаера /разные методы минимизации одно- и многовыходных двухярусных функций, методы факторизации и декомпозиции/.

Кроме программ основанных на системе Детмаера запрограммированы алгоритмы Квайна и Казакова [13] /решение проблемы покрытия множеств использует целочисленное программирование/ и алгоритм Зиссоса в котором ищутся избыточные решения без развязывания проблемы покрытия. Разработаны тоже новые алгоритмы и программы для синтеза одновыходных сетей **TANT** [14] и многовыходных трехярусных сетей **TLN** [11]. Теоретические основы этих программ представлены в работах [24, 26, 22] /сети **TLN** это генерализация многовыходных сетей **TANT**, на случай когда часть переменных только без инверсии, а часть только с инверсией/. Эти программы используют несколько эвристик независимой локальной минимизации уровней сети. Изучены некоторые стратегии подбора покрытия. Возможен тоже синтез сетей без риска [24]. Кроме того в системе существуют, запрограммированные в **LISP**: программа кодирования автомата оперта на метод Мороза, а также программа синтеза многоярусных, многовыходных комбинационных схем на вентилях **NAND**, с учетом ограничений на число входов элемента [2]. Эта программа использует в качестве подпрограммы для гене-

рации многовыходных импликантов алгоритма Шмидта - Друффеля, являющейся обобщением алгоритма Слагля. Программа выбора квазиоптимального покрытия использует множество эвристических критериев.

После этапа логического синтеза консолидирующая программа подготавливает подробное описание проектируемой системы на уровне элементарных триггеров и комбинационных элементов. В этом описании могут выступать следующие элементы **JKO** /синхронизированный триггер **JK** отключаемый передним откосом/, **JK1** /задним откосом/ **DO, D1, SRO, SR1, NAND, NOR, AND, OR, EXOR, AND-OR-INVERT, EXPANDER, NOT.**

Схема полученная таким образом может моделироваться с применением моделирующей программы основанной на компиляции, дающей возможность исследования временных явлений, выступающих на уровне логических схем. Эта программа - **MOLO** [31] делает возможным моделирование схем до 200 вентилей и 20 входов.

Описание производимое в результате структурного синтеза содержится на языке **STRUCT.** На этом языке создаются тоже конечные описания являющиеся результатом абстрактивной имплементации структуры управляющего автомата и структуры операционного автомата /цифровой части/. Данные на языке **STRUCT** являются основой для программы структурного объединенного синтеза **IMPLEM2** [36]. Эта программа вычисляет минимальные числа объединенных модулей необходимых для имплементации. Учитывает при этом возможность замена одних элементов другими. На данном этапе программа применяет целочисленное программирование /алгоритм Гоморего/. Затем программа совершает распределение элементов модуля так, чтобы сделать минимальным число соединений между модулями. Здесь отрекается условия минимального числа модулей.

5. СИСТЕМА ГРАФИЧЕСКОГО ВЫХОДА

Для системы **DIADES** разработана система графической документации, задачей которой является создание изображений графсхем алгоритмов, вспомогательных графов, блочных и логических схем.

Изображения могут печататься на строчном печатном устройстве или чертиться на присоединенном к машине регистраторе **Hewlett-Packard 9862A.** Система графической документации состоит из графического языка [27], программы размещения схем алгоритмов [33], программы размещения логических схем [37], программы трассировки соединений [32], программы квазиоптимального управления движением пишущего прибора [37], библиотеки программ - описание рисунков элементов [37, 36], программы декомпозиции графа рисунка [37] и программ смены формата данных. В настоящей версии можно нарисовать в количестве не более 64 элементов /блоков, команд, интегральных модулей/. Существование двух программ размещения элементов мотивировано разными требованиями для видов рисунка и качества размещения. Эти два алгоритма размещают рисунок сначала относительно координате **y**, затем **x**. Программа размещения графсхем ищет сначала все отдельные максимальные пути без петли, начинающиеся в инициальной вершине. Потом она генерирует на их основе ряды вершин этих колонн, учитывая разные эвристики для минимизации числа пересечений межколонных соединений. Затем чертеж размещается вдоль координаты **x** через определенный тип силового алгоритма. Программа размещения схем имеет некоторые черты силового алгоритма и программы секвенционного размещения. Программа ведения соединений использует метод **branch-and-bound** и генерирует соединения в виде квазиминимальных деревьев Стейнера. Программа эта не имеет ограничения числа дорожек в свободной полосе. Она ищет решения минимализирующего функцию оценок $p=\alpha A + \beta B + \gamma C$ причем α, β, γ являются параметрами, **A** - количеством пересечений линии, **B** - количеством загибов линии, **C** - суммарной длиной линии соединений. Алгоритм использует ряд эвристик ведения пучков соединений, выделения соединений из наконечников, и т.д. Находит он быстро хорошее соединение и имеет возможность в очередных итерациях найти соединения все лучшие. Имеет тоже программы выбора очередности соединений и создавания деревьев. Время поведения соединения составляет от 1 сек до 300 сек /**ODRA 1325**/. Соединение проведено для небольших значений α ведется примерно 300 раз быстрее чес соединения для небольших значений γ.

6. ИТОГИ

Выше описываются некоторые аспекты системы **DIADES**. Много проблем связанных с конструкцией системы, как: языки соотносительных структур, диагностика ошибок, языки описаний, каталоги, трансформация, доказательство правильности не представлены или указаны в очень сокращенном виде. Настоящая версия системы не завершена, но из ее частей уже окончено, протестировано и использовано для целей дидактики в лабораторных занятиях со студентами, посвященных теме "Автоматическое проектирование цифровых устройств". Настоящие размеры системы приблизительно следующие: трансляция - 1950 карт, абстрактная имплементация - - 500 карт, интегральная имплементация - 1200 карт, система графического выхода - 4300 карт, оптимализация и трансформация - 4700 карт, доказательство правильности и программа доказательства теорем - 2100 карт, структурный синтез 4300 карт, моделирование - 800 карт. Всего вместе ок. 20 000 карт. Опыт показывает, что трудности введения в действие системы растут быстро с ростом ее сложности. Потому автор считает в будущем необходимым построение не одной системы с неподвижной структурой, но модульной системы программирования состоящей из модулей соединенных в разные комбинации для разных целей /дидактических, научных, технических/. В эту систему будут также включены находящиеся теперь в подготовке программы: анализа отказов комбинационных схем и абстрактных автоматов /для целей синтеза бозопасных автоматов/ и программы генерирования тестов и деревьев локализации ошибок. Программы этой группы будут создавать ленты с данными для миникомпьютера **MERA 302**, на котором уже работают программы генерации тестов, тестировки и локализации отказов в цифровых устройствах подключенных в реальном масштабе времени.

Работающие части системы дают возможность многократного ускорения процесса проектирования, но эффективность применения системы ограничена батчовом трибом работы и устройствами ввода и вывода информации. Проблема эта будет частично решена после подключения к компьютеру **ODRA** миникомпьютера **MERA** в роли диалогового терминала.

ЛИТЕРАТУРА

Ниже дан только список работ непосредственно связанных с ситемой **DIADES.** Рекомендации к основной литературе по автоматическому проектированию, теории автоматов и искусственному интелекту можно найти в цитированной литературе.

1 Бачинский В.: "Сравнение методов оптимизации в пространстве эквивалентных алгоритмов". Дипломная работа исполнена в Институте Автоматики Варшавского Политехнического Института под руководством М. Перковского.
1974 /на польском языке/.

2 Банасик З.: "Структурный синтез микропрограммных автоматов в системе автоматического проектирования цифровых устройств **DIADES**". Как [1] . 1974.

3 Барановски Й.: "Организация системы автоматического проектирования". Как [1] . 1976.

4 Барановски Й.: "Язык **ADL** и его транслятор на графовый язык в системе **DIADES**". Труды **VII** Народной Конференции по Автоматике. Жешов 14-16 сентября 1977 /на польском языке/.

5 Боруславски В.: "Применение автоматического доказательства теорем для оптимализации программ". Как [1] . 1977.

6 Дворак Й.: "Глобальная оптимизация программ в системе автоматического проектирования". Как [1] . 1977.

7 Янковски К.: "Оптимизация алгоритмов в классе алгоритмов эквивалентных. Оператор организующий программную петлю в трансформированном алгоритме". Как [1] . 1975.

8 Юзвяк Л.: "Коррекция синтактических ошибок". Дипломная работа исполнена в Институте Автоматики ВПИ под руководством доц. д-р т.н. В. Трачика. 1976.

9 Юзвяк Л.: "Коррекция синтактических ошибок для языка **ADL**". Как [4] . 1977.

10 Локуцевски Р.: "Критерии выбора структуры комбинационного устройства". Как [1] . 1977.

11 Мацейчик Й.: "Синтез многовыходных трехуровневых сетей из элементов **NAND**". Как [1] . 1977.

12 Манкевич Ч.: "Выходной язык системы автоматического синтеза цифровых устройств автоматики и его транслятор на графовой язык". Как [1] . 1974.

13 Михаловски К.: "Синтез лоигических устройств с помощью вычислительной машины". Как [8] . 1972.

14 Павельска-Стэмпень Й.: "Иерархическая система автоматического синтеза комбинационных логических устройств на основе табличной репрезентации булевых функций". Как [1] . 1977.

15 Перковски М.: "Система автоматического проектирования цифровых устройств". Труды Конференции "Современные проблемы автоматики и информатики", Секция Ц, Информатика и комплексная автоматика, Гливице, 18-20 сентября 1973 г., сс. 274-305 /на польском языке/.

16 Перковски М.: "Языки относительных структур и их применение в системе автоматического блочного синтеза". Труды VI Конференции автоматики, Познань, 9-11 сентября 1974 г., т. I, с. 695-707 /на польском языке/.

17 Перковски М.: "Эвристическая оптимизация в системе автоматического блочного синтеза цифровых устройств автоматики". Труды I Симпозиума "Эвристические методы". Польское Кибернетическое Общество. Варшава, 28 сентября 1974 г., с. 101-135 /на польском языке/.

18 Perkowski M.: " A System for Automatic Design of Digital Systems". Proceedings of FCIP Symposium INFORMATICA 74 Bled Yougoslavia 7-12 October 1974, paper 4.4.

19 Перковски М., Бачински В., Янковски К.: "Эксперименты с эвристической программой для оптимизации программ". Труды II Симпозиума "Эвристические методы". Варшава, 27 сентября 1975 г., /на польском языке/.

20 Perkowski M.: "An Automated Approach to the Design of Block-Oriented Digital Systems". Institute of Automatic Control. Warsaw Technical Univ. Technical Report, 8/1975. Warsaw, October 1975.

21 Перковски М." ADL - source language of the system for automatic

design" . Труды Конференции "Организация вычислительных машин и микропрограммирование", Варшава, 24-26 IX 1975, PWN, т.I, 1976.

22 Перковски М., Рыдзевски А., Мисюревич П.: "Теория логических схем. Избранные проблемы". WPW. 1977 г., /на польском языке/.

23 Perkowski M., Jankowski K.: "Validation and Optimization of Control Automaton Programs in the System for Automatic Design". Proc. of the Symposium on Fault Diagnosis and Fault Tolerant Computing. Wisla 24-27 May 1976.

24 Perkowski M.: "An Example of Heuristic Programming Application in the Three-Level Combinational Logic Design".

Труды III Симпозиума "Эвристические методы", Варшава, 25 сентября 1976 г., с. 105-132, выпуск I.

25 Перковски М., Барановски Й.: "Проектирование цифровых устройств с применением языка ADL". Институт Автоматики ВПИ. 1976 /на польском языке/.

26 Perkowski M.: "Synthesis of Multioutput Three Level NAND Networks". Proceedings of the Seminar on Computer Aided Design. Budapest 3-5 November 1976, pp. 238-265.

27 Perkowski M.: "A Method of Validation of Parallel Programs in the System for Automatic Design of Block-Oriented Digital Systems". Proceedings of the IInd IFAC Symposium on Discrete Systems. Dresden, 14-19 March 1977.

28 Перковски М.:"Многоцелевая и многостратегическая программа решения комбинаторических задач". Труды III Симпозиума "Эвристические методы", Варшава, 25 сентября 1976 г., с. 23-90, выпуск 3 /на польском языке/.

29 Perkowski M.: "The state-space approach to the design of multipurpose problem-solver for logic design". Послано на конференцию "Artifical Intelligence and Pattern Recognition in Computer Aided Design", Grenoble, March 1978.

30 Перковски М.: "Метод пространства состояний в автоматическом синтезе логических устройств". Кандидатская диссертация. ИА ВПИ. 1977 /на польском языке/.

31 Попель Й.: "Языки для описания и моделирования цифровых устройств". Дипломная работа исполнена в ИА ВПИ под руководством к.т.н. П. Мисюревича. 1973 /на польском языке/.

32 Рудо К.: "Проблема транссировки соединений в рисунках структур цифровых устройств". Как [1] . 1974.

33 Сохонь А.: "Генерация картин графов, схем действий и блок-схем в системе автоматического проектирования". Как [1] . 1974.

34 Стравински Т.: "Метод Мазуркевича - Бликле в автоматической оптимизации программ". Как [1] . 1977.

35 Тетык Й.: "Абстрактная имплементация в системе автоматического блочного синтеза". Как [1] . 1974.

36 Томашевска К.: "Структурная имплементация на модулях **TTL SN74** в системе автоматического проектирования **DIADES**". Как [1] . 1977.

37 Урбански М.: "Система графического вывода для системы автоматического проектирования цифровых устройств". Как [1] . 1975.

38 Урбански М.: "Пакет автоматической документации логических и электронических систем". "Информатика", 1977, № 3, с. 10-12 /на польском языке/.

39 Весоловски Й.: "Трансформации алгоритмов проведения автоматов в системе автоматического блочного синтеза цифровых устройств автоматики". Как [1] . 1974.

Глава 2: ДОКЛАДЫ СИМПОЗИУМА НКС 1978 ГОДА

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1978

КЁГСТ, МАНФРЕД; ФРАНКЕ, ГЮНТЕР

К ОПИСАНИЮ АЛГОРИТМОВ ФУНКЦИОНИРОВАНИЯ ДИСКРЕТНЫХ УПРАВЛЯЮЩИХ УСТРОЙСТВ.

Академия Наук ГДР
Центральный институт кибернетики и информационных процессов. Дрезденское отделение.

I. Введение

Предпосылкой для проектирования управляющих устройств является эффективное средство описания. Из литературы известны ряд средств описания, как например таблица автомата, граф автомата, логические схемы алгоритмов ЛСА [1], граф-схемы алгоритмов ГСА [2], граф-схемы программ [3], блок-схемы программ [4,5], Organiphase [6,7] и сеть ПЕТРИ, интерпретированная в смысле коммутационной техники [8,9,10], ACDL [11] и, так называемые, Control graphs [16,17], принятые для описания управления компютером.

Представляемый в данной работе управляющий граф [12,13,14,15] содержит элементы этих средств описания. Его следует рассматривать как обобщённый граф автомата. Необходимые для асинхронного автомата условия стабильности видоизменены здесь так, что при определённом входе следует переход из одной вершины в стабильную целевую вершину, при этом могут проходить нестабильные промежуточные вершины. При этом предполагается, что или короткие импульсы на выходе при прохождении промежуточных вершин не влияют на управляемый медленный процесс, или, в другом случае, в промежуточных вершинах образуется тот же выход, как и в начальной вершине.

В противоположность к таблице автомата, в этом случае не нужно дисъюнктное описание переходных условий. Касательно возможностей описания, /например для параллельных процессов/, управляющий граф соответствует интерпретированной сети ПЕТРИ. Но для описания параллельной работы здесь используют переменные вершин как при Organiphase [6,7].

Управляющий граф разработан для описания, анализа и моделирования асинхронных проблем управления на уровне описания. Он может служить и основой для преобразования на уровне описания и для простых трансформаций в логические схемы или программы для управляющих компютеров.

2. Концепт графа управления

2.1. Основные положения.

Суть описания состоит в тройке:

Ситуация - Условие - Последовательная ситуация

Исходя из грубого описания при помощи комплектных ситуаций и условий возможно любое уточнение путём разделения ситуаций или условий на дальнейшие тройки. Ситуациям соответствуют вершины, а условиям - рёбра ориентированного графа.

Если управляющая система находится в ситуации i, что значит, что вершина Z_i маркирована, то она при выполнении определённых условий переходит в ситуацию j, что значит, что марка от Z_i переходит к Z_j. В случае параллельной работы одновременно маркированы несколько вершин.

Функции выхода, которые могут быть и автоматными отображениями, привязаны к вершинам. В качестве условий могут выступать статические условия, условия автоматного отображения, динамические условия, условия времени и глобальные условия.

В нижеследующем определении управляющего графа исходят из того, что как автоматные отображения выхода в вершинах, так и многообразие условий рёбер могут быть описаны с помощью частичных булевых функций переменных входов и вершин.

Для описания асинхронных управляющих систем с n входных переменных $x_1,\ldots,x_n$ и m выходных переменных $y_1,\ldots,y_m$ вводят управляющий граф.

Определение: Управляющий граф /УГ/ о множестве всех частичных булевых функций от (n+k) переменных является ориентированный, не обязательно взаимозависимый, по вершинам и рёбрам взвешенный граф $S = (Z,K,v,w)$

где $Z=\{Z_1,\ldots,Z_k\}$ - множество вершин

$K, K \subseteq Z \times Z$ - множество рёбер

$v: Z \rightarrow G^m$ - вес вершин

$w: K \rightarrow G$ - вес рёбер

$S=(Z,K,v,w,Z^{(o)})$ с $Z^{(o)}$ из $\mathfrak{H}'(Z)$ значит инициальный УГ.
Здесь $\mathfrak{H}(Z)$ - множество всех подмножеств от Z.
Для удобства описания в дальнейшем каждому элементу $Z^{(r)}$ из $\mathfrak{H}(Z)$ соответствует и одновременно допустим набор
$M^{(r)}=(\sigma_1,\ldots,\sigma_k)$ с $\sigma_i = \begin{cases} 0, \text{для} & Z_i \notin Z^{(r)} \\ 1, \text{для} & Z_i \in Z^{(r)} \end{cases}$
Как и $M^{(r)}$ в сети ПЕТРИ, так и $Z^{(r)}$ обозначаются в качестве маркировки. При этом условии G можно интерпретировать следующим образом:

$$\{0,1\}^n \times \mathfrak{H}(Z) \rightarrow \{0,1\} .$$

Способ записи и спецификация взвешиваний:
Вершина Z_i из Z взвешивается с помощью выходных функций

$$v(Z_i) = (g_{(i),1},\ldots,g_{(i),m})$$

Для каждого $\mu, 1 \leq \mu \leq m$ булевая функция $g_{(i),\mu} = (D^0_{(i),\mu}, D^1_{(i),\mu})$ определяется множеством $D^0_{(i),\mu}$ или $D^1_{(i),\mu}$ её нуливых или единичных наборов. Вес ребра (Z_i,Z_j) из К обозначен $w(Z_i,Z_j) = f_{ij}$,
где функция $f_{ij}=(D^0_{ij},D^1_{ij})$ задана множеством D^0_{ij} или D^1_{ij} её нулевых или единичных наборов. При этом предполагается, что для выдачи $v(Z_i)=(g_{(i),1},\ldots,g_{(i),m})$ в вершине Z_i и для эффективности функции $w(Z_i,Z_j)=f_{ij}$ ребра от Z_i до Z_j вершина Z_i маркирована. Из соответствующих рёбрам локальных функций перехода f_{ij} и соответствующих вершинам локальных функций выхода $(g_{(i),1},\ldots,g_{(i),m})$ в УГ можно получить описание общего поведения в смысле теории автоматов с помощью глобальной функции перехода f и глобальной функции выхода g.
При маркировке $Z^{(r)}$ получается общий выход g как дизъюнкция локальных выходов всех маркированных вершин.
Для инициального УГ $S=(Z,K,v,w,Z^{(o)})$, описывающего конкретную управляющую систему, должно быть удовлетворено требование: соответствующие локальные выходы для всех от начальной маркировки $Z^{(o)}$ достигаемых маркировок свободны от противоречий.
Кроме этого, при описании конкретных управляющих систем часто имеет смысл модифицирование условий устойчивости, обычных для асинхронных автоматов, как следует:

$$\forall Z_j \forall \sigma \, (Z_j \in Z \wedge \sigma \in \bigcup_{l \in N(j)} D_{jl} \longrightarrow E(Z_j,\sigma) \neq \emptyset).$$

При этом вершина Z_k элемент от $E(Z_j,\sigma)$, если выполняется:

$$\exists d \exists \{Z_{j1},\ldots,Z_{jd}\}\ (d\in Nz \wedge \{Z_{j1},\ldots,Z_{jd}\}\subset Z \wedge Z_j = Z_{j1}$$
$$Z_k = Z_{jd} \wedge \forall\delta(\delta\in\{1,\ldots,d-1\} \longrightarrow \sigma\in D_{j_\delta j_{\delta+1}}) \wedge \sigma \notin \bigcup_{l\in N(k)} D_{kl}).$$

Модифицированное условие допускает возможность перехода от одной вершины при наборе переменных через неустойчивые вершины к устойчивой вершине. Из этого следует, что для общего выхода короткие импульсы, возникающие при прохождении промежуточных вершин, не вызывают ошибочного поведения в управляемом процессе.

Соглашения:

Для вершины Z_j УГ $S=(Z,K,v,w)$ множество индексов предыдущих вершин от Z_j обозначается:

$$V(j) = \{i \,/\, (Z_i,Z_j)\in K \wedge i\neq j\}$$

множество индексов последующих вершин от Z_j как:

$$N(j) = \{k \,/\, (Z_j,Z_k)\in K \wedge j\neq k\}$$

Так как вес $f_{ij}=(D^o_{ij},D^1_{ij})$ ребра (Z_i,Z_j) должен выполнять условие:

$$(D^1_{ii} \cup \bigcup_{k\in N(i)} D^1_{ik}) \setminus D^1_{ij} \subseteq D^o_{ij}$$

в дальнейшем без ограничения общности предположим:

$$(D^1_{ii} \cup \bigcup_{k\in N(i)} D^1_{ik}) \setminus D^1_{ij} = D^o_{ij}$$

Поэтому достаточно указать для веса ребра $w(Z_i,Z_j)$ вместо $f_{ij}=(D^o_{ij},D^1_{ij})$ только D^1_{ij}. Так как в дальнейшем будет использоваться только D^1_{ij} верхний индекс можно опустить и писать D_{ij} для D^1_{ij}. D_{ij} обозначает область определения перехода. Множество D_{ij} содержит технологически возможные наборы переменных, для которых переходят от Z_i к Z_j. Соответственно D_j обозначается множество выступающих в вершине Z_j наборов областью определения вершин. Она является объединением областей определения переходов рёбер ведущих в эту вершину, то есть:

$$D_j = \bigcup_{i\in V(j)} D_{ij} \cup D_{jj}$$

Для практической записи УГ D_j включается в описание. В результате этого запись петель (Z_j,Z_j) в УГ и явная запись их областей определения D_{jj} становятся лишними.

Практическая запись управляющего графа:

УГ $S=(Z,K,v,w)$ может быть описан путём указания для каждой вершины Z_j из Z веса $v(Z_j)=(g_{(j),1},\ldots,g_{(j),m})$, а для каждого ребра (Z_j,Z_k) с $j\neq k$ веса D_{jk}. В дальнейшем в основу описания УГ будет положен этот вид описания. При этом вершина Z_j для общего случая будет охарактеризована следующими параметрами:

1) D_j
2) $v(Z_j)=(g_{(j),1},\ldots,g_{(j),m})$ с $g_{(j),\mu}=(D^0_{(j),\mu},D^1_{(j),\mu})$ для $\mu = 1,2,\ldots,m$
3) $V(j)$ и $N(j)$
4) D_{ij} для всех i из $V(j)$ и D_{jl} для всех l из $N(j)$.

Эта общая форма записи УГ функционально ориентирована. Она допускает детальное описание задачи и является основой для анализа и преобразований. Но функции могут быть заменены и с помощью выражений:

- вес рёбер f_{ij} обозначим, так называемым, выражением включения H_{ij}, а
- вес вершин $(g_{(i),1},\ldots,g_{(i),m})$ -кой выражений выходов.

При этом следует обратить внимание на то, что при вычислении выражений включения и выходов определяют неопределённые места в соответственных функциях, так что нельзя больше перейти от выражений к функциям. Поэтому для определения функций из выражений необходима дополнительная информация о наборах, встречающихся в управляемом процессе. Описанный с помощью выражений включения и выходов УГ реализовано ориентирован. Он является основой для непосредственной трансформации в логические схемы или программы для управляющих компютеров. Кроме этого для удобства описания возможно одновременное употребление функций /или областей определения/ и выражений. /рис.I/ Однако при разработке задачи чаще начинают с записи выражений включения и выходов.

Для описания несинхронного управления с помощью УГ за основу берётся следующая интерпритация: Возможным в управлении ситуациям соответствуют, как ранее было согласовано, вершины, а переходам между ситуациями - рёбра УГ. Область определения ребра D_{ij} содержит наборы переходов из ситуации Z_i в ситуацию Z_j.

Примечание: В случае, если УГ должен быть проверен на надёжность в смысле сети ПЕТРИ, следует считаться с нормированной областью определения вершин.

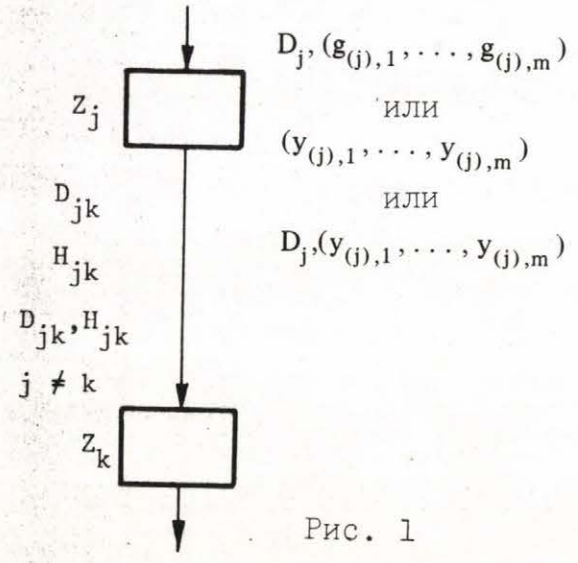


Рис. 1

3. Преобразование управляющих графов

Преобразование УГ имеет смысл для достижения следующих целей: снижение затрат на техническое и программное обеспечения, снижение объёма документации, уменьшение времени вычисления на ЭВМ, экономия затрат памяти, адаптация структуры к стандартному блоку. Существует возможность преобразования УГ с помощью объединения вершины с вершиной, или вершины с ребром, или разъединения вершин с введением параллельной работы или без неё. Однако с точки зрения разработки конкретных управлений рекомендуется соединять по возможности только соседствующие вершины, чтобы получить обозримую структуру соответствующего управляющего графа и его последующую реализацию. Это имеет преимущества для производства, обслуживания, ремонта и возможного дополнительного корректирования задачи. Здесь указываются только условия для объединения вершин, которые встречаются чаще всего.

Условием для объединения вершин является непротиворечивое описание задачи с помощью УГ, причём допускаются прохождения и параллельные работы.

Для описания условий совместимости для функции $f_{ij}(x_1,\ldots,x_n,z_1,\ldots,z_n)$ или функций выходов $g_{(i)},(x_1,\ldots,x_n,z_1,\ldots,z_n)$ переменных входа $x_1,\ldots,x_n$ и переменных вершины $Z_1,\ldots,Z_n$ требуются те продолжения $f'_{ij}(x_1,\ldots,x_n)$ или $g'_{(i),\mu}(x_1,\ldots,x_n)$, которые зависят только от переменных входа. Аналогично здесь обозначается и область определения рёбер как D'_{ij}, а область определения вершин как D'_j Для таких УГ, или их частей, в которых не встречается параллельная работа, достаточно рассматривать функции выхода и перехода только как функции переменных входа $x_1,\ldots,x_n$ и считаться с штрихованными областями определения D'_{ij} и D'_j. Итак, обозначаются параметры без параллельной работы штрихом, как например $D'_{ij}, g'_{(j),1}$.

Условие для совместимости двух вершин:

Две вершины Z_j и Z_k при $j \neq k$ постояно объединимы, если они выступают одновремено как маркированные или как немаркированные.

Две вершины Z_j и Z_k при $j \neq k$, которые никогда одновремено не маркированы могут объединятся /рис.2/, $Z_p = Z_{j+k}$

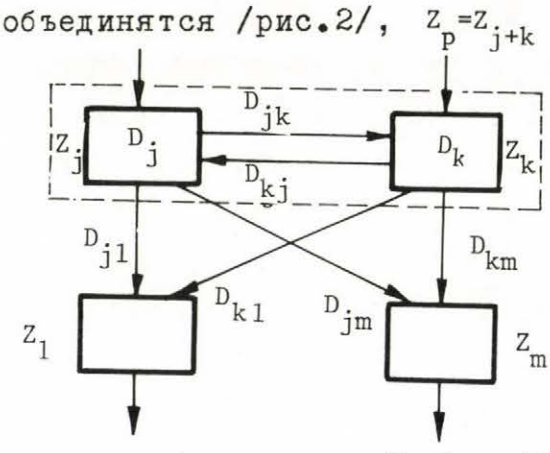


Рис. 2

если:

I. Z_j /или Z_k/ является предшественником Z_k/или Z_j/, и не существует предшественник Z_i от Z_k /или Z_j/, который отличался бы от Z_j, и одновременно был бы маркирован с Z_j /или Z_k /.

2. Z_j и Z_k совместимы касательно выдачи, то есть по компонентам функции выдачи $(g_{(j),1},\ldots,g_{(j),m})$ в Z_j и $(g_{(k),1},\ldots,g_{(k),m})$ в Z_k. $D'_j D'_k = \emptyset$ является для этого достаточным условием. Если оно не выполняется, то следует проверить выполняется ли $g_{(j),\mu}(\tau) = g_{(k),\mu}(\tau)$ для каждого набора из $D'_j D'_k$, если обе величины определены.

3. Z_j и Z_k совместимы касательно продолжения, то есть, при их объединении не должен возникать нежелательный переход из Z_j или Z_k. Достаточным условием является то, что для каждой последующей вершины $l \in N(j) \setminus \{k\}$ выполняется $D'_{jl}D'_k \subseteq D'_{kl} \cup D'_{kj}$ и соответственно для каждого $m \in N(k) \setminus \{j\}$ - $D'_jD'_{km} \subseteq D'_{jk} \cup D'_{jm}$.

4. При объединении Z_j и Z_k не возникает параллельная работа. Достаточным условием является то, что для каждой пары (l,m) последующих вершин с $l \in N(j) \setminus \{k,m\}$ и $m \in N(k) \setminus \{j,l\}$ выполняется $D'_{jl}D'_{km} \subseteq (D'_{jk} \cup D'_{jm})(D'_{kj} \cup D'_{kl})$.

Условия 3 и 4 могут быть представлены следующим образом:

$$\forall l(l \in (N(j) \cup N(k)) \setminus \{j,k\} \longrightarrow (D'_{jl} \cup D'_{kl}) \cap$$

$$\cap ((D'_k \cup \bigcup_{i \in N(k)} D'_{ki}) \setminus (D'_{kj} \cup D'_{kl})) \cup ((D'_j \cup \bigcup_{i \in N(j)} D'_{ji}) \setminus (D'_{jk} \cup D'_{jl}))) = \emptyset.$$

Если две вершины Z_j и Z_k выполняют условия совместимости, то они могут быть объединены в новую вершину $Z_p = Z_{j+k}$. Для постепенного продолжения объединения с новой вершиной Z_p следует определить её параметры.

Для того случая, если управляющая система не содержит параллельной работы, расчёт параметров для Z_p /параметры от $Z_p = Z_{j+k}$ без параллельной работы/, проведён следующим образом:

I. $D'_p = D'_j \cup D'_k$

2. $v(Z_p) = (g'_{(p),1}, \ldots, g'_{(p),m})$ $g'_{(p),\mu} = (D^{0'}_{(p),\mu}, D^{1'}_{(p),\mu})$

   и $D^{0'}_{(p),\mu} = D^{0'}_{(j),\mu} \cup D^{0'}_{(k),\mu}$ и $D^{1'}_{(p),} = D^{1'}_{(j),\mu} \cup D^{1'}_{(k),\mu}$

   или $\mu = 1, \ldots, m$

3. $V(p) = (V(j) \cup V(k)) \setminus \{j,k\}$ и $N(p) = (N(j) \cup N(k)) \setminus \{j,k\}$

4. $D'_{ip} = D'_{ij} \cup D'_{ik}$ для всех i из V(p) и

   $D'_{pl} = D'_{jl} \cup D'_{kl}$ для всех l из N(p).

Для определения параметров от $Z_p = Z_{j+k}$ -/спараллельной работой/ следует учитывать переменные вершин /нештрихованные функции и области определения/, число которых при объединении уменьшается на единицу. Правила для определения параметров от Z_p /без пераллельной работы/ применимы и для определения параметров от Z_p /с параллельной работой/ при соблюдении следующего:

В каждой функции выходов и каждой области определения вершин и рёбер следует записать переменную вершины Z_p вместо опущенных переменных вершин Z_j и Z_k; при этом в каждом наборе значение I придаётся Z_p только тогда, когда Z_j или Z_k имело значение I.

4. Реализация УГ

В основу реализации УГ в качестве логической схемы и программы положены выражения включения и выходов. После окончания преобразований эти выражения могут быть рассчитаны из соответственных функций перехода и выхода с применением известных методов минимизации. Выражение включения H_{ij} к ребру (Z_i, Z_j) получается из булевой функции $f_{ij}=(D^0_{ij}, D^1_{ij})$ с

$$D^0_{ij} = (D_i \cup \bigcup_{k\in N(i)} D_{ik}) \setminus D_{ij} \text{ и } D^1_{ij} = D_{ij}.$$

Точно также можно рассчитать выражение выхода μ-ого выхода для i-ой вершины из функций выходов

Общий выход УГ можно тогда указать как дизъюнкцию выхода маркированных вершин:

$$Y = \bigvee_{Z_j\in Z} (y_{(j),1}, \ldots, y_{(j),m})^{z_j}$$

Для многократного использования логических элементов расчёт общего выхода может быть сведён к расчёту булевых функций.

При реализации логической схемы самое простое и обозримое по структуре преобразование состоит в том, что каждой вершине УГ соотносят памятный элемент. Логику для включения памятных элементов получают из необходимых для включения предшествующих вершин и соответственных выражений включения. При условии, что в УГ для каждого набора из D_{ij} вершина Z_j не маркирована, и имеющаяся в вершине Z_i марка передаётся вершине Z_j, возможно достижение обратного включения памятных элементов ставшими маркированными следующими вершинами, /при известных условиях в связи с соответственными выражениями включения/. Но реализация возможна и с меньшим числом памятных элементов путём соотнесения каждой вершине кодового значения, то есть набора памяти. Однако при этом следует обратить внимание на то, что кодирование свободно от состязания. Кроме этого, здесь могут быть потеряны преимущества обозримости и удобного дополнительного корректирования. Другая возможность состоит в

применении стандартной структуры для реализации УГ /I8/. Например [16,17] останавливаются на реализации УГ в качестве программы для управляющего компютера.

5. Пример:

На этом простом примере показывается описание и преобразование УГ. Две вагонетки W_1 и W_2 едут, как это показано на рис.3 [19], между контактами x_I и x_2 или x_3 и x_4. Переменные выходов y_I или y_3 направляют движение вагонеток W_1 или W_2 вправо, а y_2 или y_4 - влево. При старте обе вагонетки находятся слева $x_I = x_2 = I$ Кнопка x_5 приводит обе вагонетки в движение на право. W_1 возвращается после достижения контакта x_2, а W_2 ожидает у контакта x_4 пока W_1 достигнет контакт x_I. УГ с выражениями включения и выхода /без областей определения/ показан на рис.4. В вершинах записаны только выходы со значением I. Этот УГ может непосредственно служить основой для преобразования в логическую схему или программу.

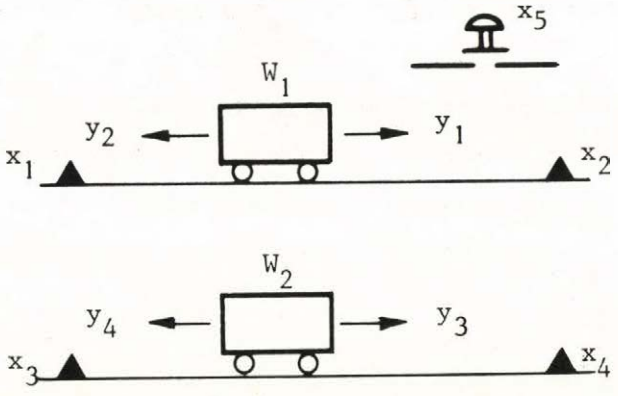


Рис. 3

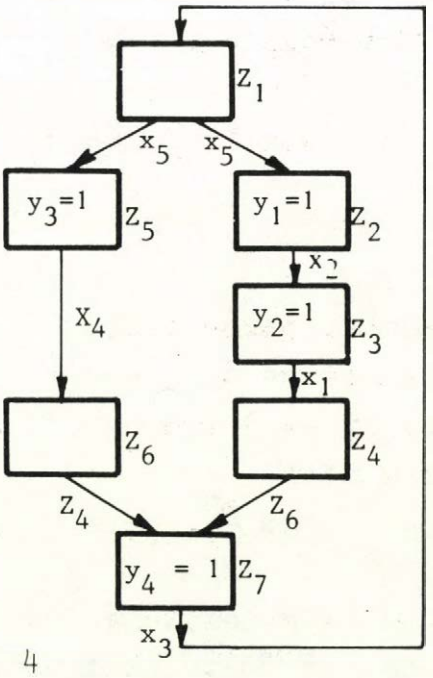


Рис. 4

Если желательны преобразования, тогда надо записать /локально/ для соответственных вершин и исходящих из них рёбер возможные наборы входных переменных, /то есть области определений/ с помощью уточняющих запросов к управляемому процессу. При этом необходимо определять только наборы тех переменных, которые изменяются в области соответственных вершин. Показывается объединение вершин Z_3 и Z_4. Вначале записываются необходимые области определения:

$$\begin{array}{lccccc} & x_1 & x_2 & x_3 & x_4 & x_5 \\ D'_3 = \{ & 0 & - & - & - & - \} \\ D'_4 = \{ & 1 & 0 & - & - & - \} \\ D'_{47} = \{ & 1 & 0 & 0 & 1 & - \} \end{array}$$

Следует проверка условий совместимости Z_3 и Z_4 , это значит $j = 3$ и $k = 4$ Первое условие выполнено;
Второе условие - совместимость функций выдачи - выполнено, так как $D'_3 D'_4 = \emptyset$; Третье условие - совместимость продолжений выполено, так как $D'_3 D'_{47} = \emptyset$; Четвертое условие возникновение нежелательной параллельной работы - отпадает, так как Z_3 и Z_4 не имеют последующих пар.
Следовательно вершины Z_3 и Z_4 совместимы и объединяются в Z_{3+4}.Области определения /без параллельной работы/ для вершины Z_{3+4} и ребра от Z_{3+4} до Z_7 получают как следует:

$$\begin{array}{lccccc} & x_1 & x_2 & x_3 & x_4 & x_5 \\ D'_{3+4} = D'_3 \cup D'_4 = \{ & 0 & - & - & - & - \\ & 1 & 0 & - & - & - \} \\ D'_{3+4,7} = D'_{4,7} = \{ & 1 & 0 & 0 & 1 & - \} \end{array}$$

Рёбра, ранее ведущие в Z_3, ведут теперь в Z_{3+4}. Выражение включения $H_{3+4,7}$ для перехода из Z_{3+4} в Z_7 можно рассчитать по булевой функции $f_{3+4,7} = (D^0_{3+4,7}, D^1_{3+4,7})$
при этом:

$$D^0_{3+4,7} = D_{3+4} \setminus D_{3+4,7} \qquad \text{и} \quad D^1_{3+4,7} = D_{3+4,7}$$

При этом получают нештрихованные области определений с помощью дополнительных переменных Z_{3+4} и для соответственных вершин Z_{3+4} и Z_6

$$D_{3+4} = \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 & z_{3+4} & z_6 \\ \{\ 0 & - & - & 0 & - & 1 & 0 \\ 0 & - & 0 & 1 & - & 1 & 1 \\ 1 & 0 & - & 0 & - & 1 & 0 \\ 1 & 0 & 0 & 1 & - & 1 & 1\ \} \end{matrix}$$

$$D_{3+4,7} = \{\ 1\ 0\ 0\ 1\ -\ 1\ 1\ \}$$

Отсюда следует для $D_{3+4,7} = D_{3+4} \setminus D_{3+4,7}$

$$D^0_{3+4,7} = \begin{matrix} \{\ 0 & - & - & 0 & - & 1 & 0 \\ 0 & - & 0 & 1 & - & 1 & 1 \\ 1 & 0 & - & 0 & - & 1 & 0\ \} \end{matrix}$$

Отсюда получают два решения для выражения включения:

$$H_{3+4,7} = x_1x_4 \quad \text{или} \quad x_1z_6$$

Выражение $y_{(3+4),2}$ второго выхода в вершине Z_{3+4} следует из выходной функции: $g_{(3+4),2}=(D^0_{(3+4),2}, D^1_{(3+4),2})$ с

$D^0_{(3+4),2} = \{1\ 0\ -\ -\ -\ 1\ -\}$ и

$D^1_{(3+4),2} = \{0\ -\ -\ -\ -\ 1\ -\}$ к $y_{(3+4),2}=x_1$

Полученный УГ показан на рис.5.

При достаточном опыте проектировщика не нужна такая подробная поэтапная запись обьединения вершин. В примере возможно и объединение вершин Z_5 и Z_6 в вершину Z_{5+6}, и вершин Z_1 и Z_7 в Z_{1+7}. Полученный при этом УГ показан на рис.6.

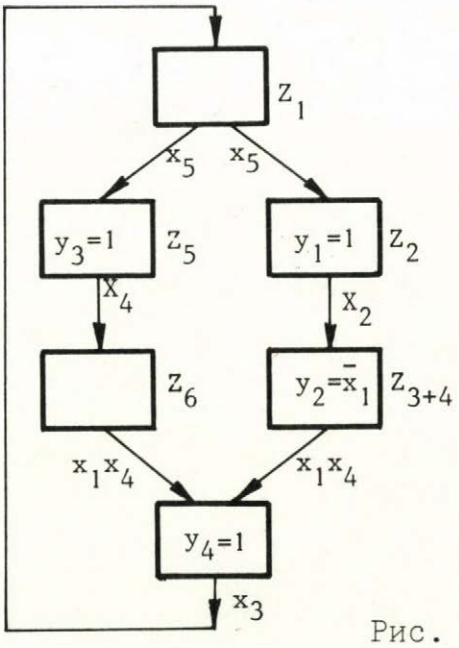


Рис. 5

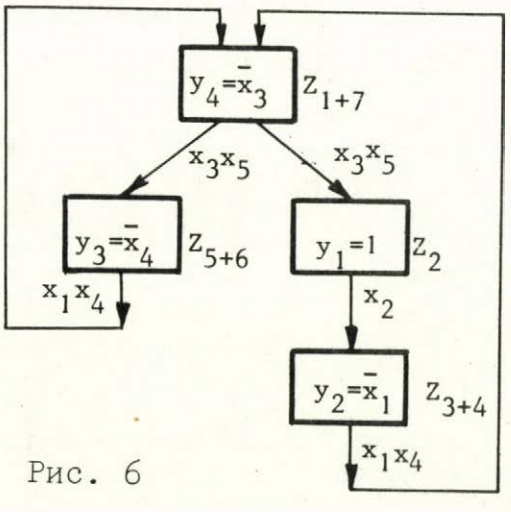


Рис. 6

6. Литература:

/1/ Djatschenko,W.F.; Lasarew,W.G.
Umformungen logischer Algorithmenschemata und Vereinfachung der Struktur von Mikroprogramm-Automaten
EIK 4(1968)H.3, 173-186

/2/ Баранов, С.И.;
Синтез микропрограммных автоматов
Ленинград: Энергия 1974г.

/3/ Killenberg,H.
Verhaltensbeschreibung von Schaltsystemen mit Hilfe von Programmablaufgraphen
msr 19(1976)H.6, 197-201

/4/ Backes,H.W.;Neulist,K.;Schaffernak,K.A.
Der Programmablaufplan - ein Mittel zur Behandlung festverdrahteter Prozeßsteuereinrichtungen
BBC-Nachrichten 51(1969)H.12, 674-680

/5/ Habiger,E.; Heß,K.
Der Programmablaufplan - ein effektives Arbeitsmittel für die Planung, Projektierung, Beschreibung, Prüfung, Inbetriebnahme und Wartung industrieller Steuerungen.
Der VEM-Elektro-Anlagenbau 12(1976)H.1, 1-9

/6/ Prunet,F.; Dumas,J.M.
A convenient tool for designing, analysing and synthesing logical sequential systems
IFAC-Symp. Discrete Systems Riga 1974

/7/ Dumas,J.M.; Prunet,F.
Searching for properties on a parallel processes model.
Proceedings of the 1975 IFAC-IFIP Workship on Real-time programming
Boston/Cambridge/Massachusetts 1975

/8/ Winkowski,J.
Formal theories of Petri nets and net simulation
CC PAS Report 242. Warschau 1976

/9/ König,R.
Petri nets and their Application for a Standardizable Design of Switching Circuits
IFAC-Symp. Discrete Systems Dresden 1977

/10/ Wendt,S.
Using Petri nets in the design process for interacting asynchronous sequential circuits
IFAC-Symp. Discrete Systems Dresden 1977

/11/ Bednar,G.M.; Tracey,J.H.
An Asynchronous Circuit Design Language (ACDL)
IEEE Transact. on Comp. (1974)No.9, 971-976

/12/ Franke,G.; Koegst,M.
Ein Verfahren zur Zustandsreduktion für taktkettenähnliche Steuerungen
Int.Wiss.Kolloquium, TH Ilmenau 1976

/13/ Franke,G.; Koegst,M.
Zustandsverschmelzung in verallgemeinerten (nichtdisjunkten) Automatengraphen
IFAC-Symp. Discrete Systems Dresden 1977

/14/ Oberst,E.; Kieser,M.; Nauber,P.
An approach to a quick and information-intensive description of digital industrial switching problems
IFAC-Symp. Discrete Systems Dresden 1977

/15/ Oberst,E.; Koegst,M.; Franke,G.
Beschreibung binärer Steuerungen durch Steuergraphen
msr 21(1978)H.10, 572-578

/16/ Despang,G.
Zur Softwarearchitektur speicherprogrammierter Steuerungen
IFAC-Symp. Discrete Systems Dresden 1977

/17/ Despang,G.
Programmierung von Mikrorechnern in der Steuerungstechnik auf der Grundlage von Steuergraphen
Elektrie 1979

/18/ Hummitzsch,P.; König,R.
Corex - ein System zum Entwurf komplexer digitaler Systeme
ZKI-Informationen 1978

/19/ Silva-Suarez,M.; David,R.
On the programmation of asynchronous sequential systems by logic equations
IFAC-Symp. Discrete Systems Dresden 1977,Bd.2,120-129

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1978

ON FORMALIZED REPRESENTATION OF NONDETERMINISTIC PARALLEL PROCESSES

GERBER S., HAUBOLD K.
Karl-Marx-Universität Leipzig, Sektion Mathematik

ABSTRACT

In this paper is explained the characteristic for a language, which can be used for the description of the structure of computation and control processes. By special elements of the language it is possible to declare subroutines in various levels, nondeterministic and parallel processings. It is stated a well defined interpretation of the syntax and a connection to the graph schemes notion.

1. Let V and W be nonempty, enumerable sets (called set of variables and set of values resp.) By W^V we mean the set of all (partial) functions (called valuations) from V into W and $e \in W^V$ is the function with empty domain, i.e. $De = \emptyset$. Moreover we note $v \in V$, $V' \subseteq V$, $b, b_i \in W$ and $B, B_i \in W^V$.
For the valuations from W^V we define:

1° The valuation b_2 is a *covering* of b_1 *in reference to* V' (abbr. $b_1 \leqslant_{V'} b_2$), if $v \in Db_1$ implies $v \in Db_2$ and $b_1(v) = b_2(v)$ for all $v \in V'$.

2° The valuations b_1 and b_2 are *equal in reference* to V' (abbr. $b_1 =_{V'} b_2$), if $b_1 \leqslant_{V'} b_2$ and $b_2 \leqslant_{V'} b_1$.

3° The *supplement* of b_2 by b_1 (abbr. $b_1 \cdot b_2$) is defined as follows:

$$b_1 \cdot b_2\,(v) =_{df} \begin{cases} b_1(v) \\ b_2(v) \end{cases}$$

if $v \in Db_1$ and $v \notin Db_2$
$v \in Db_2$

As is easy to see, that the following properties are valid:

- The relation $\underset{V'}{\leqslant}$ is reflexive and transitive.
- $\underset{V'}{=}$ is an equivalence relation.
- $(W^V, \cdot, e)$ is a monoid with unit e.
- $b_1 \underset{V''}{\leqslant} b_2$ implies $b_1 \underset{V'}{\leqslant} b_2$ for all $b_1, b_2 \in W^V$ and $V' \subseteq V''$.

Now we expand the notion introduced above to sets of valuations:

1^0 $B_1 \underset{V'}{\leqslant} B_2$ if and only if $\forall b_1 (b_1 \in B_1 \rightarrow \exists b_2 (b_2 \in B_2 \wedge b_1 \underset{V'}{=} b_2))$

2^0 $B_1 \underset{V'}{=} B_2$ if and only if $B_1 \underset{V'}{\leqslant} B_2$ and $B_2 \underset{V'}{\leqslant} B_1$

3^0 $B_1 \cdot B_2 \underset{df}{=} \{b_1 \cdot b_2 \;/\; b_1 \in B_1 \text{ and } b_2 \in B_2\}$

$$b \cdot B \underset{df}{=} \{b\} \cdot B, \qquad B \cdot b \underset{df}{=} B \cdot \{b\}$$

We denote, that for this are valid also the above mentioned properties.
The unit of the monoid is now the set $\{e\}$.
By the multiple supplement we shall understand the following operation:

$$\prod_{i=1} b_i \underset{df}{=} b_1, \qquad \prod_{i=1}^{n+1} b_i \underset{df}{=} \left(\prod_{i=1}^{n} b_i\right) \cdot b_{n+1}$$

and respectively for sets of valuations:

$$\prod_{i \in \emptyset} B_i \underset{df}{=} \emptyset, \qquad \prod_{i=1} B_i \underset{df}{=} B_1, \qquad \prod_{i=1}^{n+1} \underset{df}{=} \left(\prod_{i=1}^{n} B_i\right) \cdot B_{n+1}.$$

Let F be the set of all functions from W^V into 2^{W^V} and $f \in F$. Such functions arises in execution of the programs introduced later. For subsets B of W^V we write $f(B)$ instead of $\bigcup_{b \in B} f(b)$. At last we shall introduce the relations including and equivalence for functions from F.

1^0 The function f_1 is said *included* in f_2 *in reference to* V' (abbr. $f_1 \underset{V'}{\leqslant} f_2$), if $b \in Df_1$ implies $b \in Df_2$ and

$f_1(b) \leqslant_{V'} f_2(b)$ for all $b \in W^V$.

2^0 The functions f_1 and f_2 is said *equivalent in reference to* V' (abbr. $f_1 =_{V'} f_2$), if $f_1 \leqslant_{V'} f_2$ and $f_2 \leqslant_{V'} f_1$.

It is easy to show, that $\leqslant_V$, is reflexive and transitive, and $=_V$, is an equivalence relation.

2. Now we offer the syntax of the language, which will be defined over the set of words W(X), whereat the alphabet X may include the set of variables V and different to them the signs : , ; , ' , → , / , (, [, < ,) ,] , > , ~ , V, Λ , I , Z , 1 , 0. We assume that W(X) is a subset of W. We understand by $\underline{T}$ the set of *terms* from a sublanguage. These terms describes the application of the elementary operations, for ex. an in [3]. The set $\underline{L}$ of *logical terms* let be a subset of $\underline{T}$. Following properties may be valid for these terms and logical terms:

- The signs :,;,',/,→,I,Z are not present in $\underline{T}$.
- $t \in \underline{T}$ implies $n_x(t) = n_y(t)$ with x resp. y from {(,[,<} resp. {),],>}. [1]
- $\{1,0\} \subseteq \underline{L}$, $\ell, \ell_1, \ell_2 \in \underline{L}$ implies $\tilde{}\ell, (\ell_1 \vee \ell_2), (\ell_1 \wedge \ell_2) \in \underline{L}$.
- The set $\underline{T}$ is syntactical unambigious.

Under these conditions we introduce now the set $\underline{PT}$ (resp.$\underline{P}$) of the *program terms* (resp. *programs*) as follows:

1^0 $\underline{PT}_0 \overset{df}{=} \{I, Z, /v/, v::t \;/\; v \in V, t \in \underline{T}\}$, $\underline{P}_0 \overset{df}{=} \underline{PT}_0$

2^0 $\underline{PT}_{i+1} \overset{df}{=} \{p_t, v{:}p_t, \ell \to p_t, <\ell,p>, (q), [q], v[p_1,p_2] \;/$

$p_t \in \underline{PT}_i$, $v \in V$, $\ell \in \underline{L}$, $p, p_j \in \underline{P}_i$, $q \in P_i \cdot W(\{,\} \cdot \underline{P}_i)$

$\underline{P}_{i+1} \overset{df}{=} \{p, p_t, p;p_t \;/\; p \in \underline{P}_i, p_t \in \underline{PT}_i\}$

[1] $n_x(t)$ denotes the number of places with x in t.

$$3^0 \quad \underline{PT} \underset{df}{=} \bigcup_{i \in Int.} \underline{PT}_i \quad , \quad \underline{P} \underset{df}{=} \bigcup_{i \in Int.} \underline{P}_i$$

With such program terms from $\underline{PT}$ (resp. programs from $\underline{P}$) the structure of algorithms will described syntactically. For any application the set $\underline{T}$ must be choosen suitably.

For program terms resp. programs following properties are current:

- $\underline{P} \cap \underline{T} = \emptyset$, $\underline{PT} \subset \underline{P}$,
- $\underline{PT}$ and $\underline{P}$ are syntactical unambigious,
- For every $p \in \underline{P}$ exists one and only one number $n_p \geq 1$, and one function $\varphi_p : \{1,\ldots,n_p\} \to \underline{PT}$, with $p \underset{df}{=} \varphi_p(1);\ldots;\varphi_p(n_p)$

Every program arises by concatenation of program terms and is vica versa unambigiously decomposable in these program terms. Therefore the set $\underline{P}$ is free generable from $\underline{PT}$ together with the seperator ; .

3. For the interpretation of program terms and programs we now introduce the function Val from the set $\underline{P} \times W^V$ into 2^{W^V}. We presume that exist an interpretation (signed also with Val) for the terms from $\underline{T}$ and the logical terms from $\underline{L}$, i.e. the function Val is a (partial) mapping with

$$\underline{T} \times W^V \to W \ , \quad \underline{L} \times W^V \to \{true, false\}, \quad \underline{P} \times W^V \to 2^{W^V} \ .$$

Further we demand, that by Val the signs 1,0 and the functors $\sim, \vee, \wedge$ are interpreted as *true*, *false* and as negation, disjunction, conjunction respectively.

The interpretation of the programs is then defined so:[1)]

$$1^0 \quad Val(I,b) \underset{df}{=} \{e\} \ , \ Val(Z,b) \underset{df}{=} \emptyset \quad \text{for all} \quad b \in W^V$$

$$Val(/v/,b) \underset{df}{=} Val(b(v),b) \quad \text{if} \quad b(v) \in \underline{P}$$

$$Val(v::t,b) \underset{df}{=} \{b'\} \quad \text{with} \quad b'(v) \underset{df}{=} Val(t,b) \quad \text{and} \quad Db' \underset{db}{=} \{v\}$$

1) The value of the left hand program is only then defined, if the right hand site of the equation is defined completely.

2^0 $\mathrm{Val}(v{:}p_t,b) \underset{df}{=} \{b!\}$ with $b'(v) \underset{df}{=} p_t$ and $Db' \underset{df}{=} \{v\}$

$$\mathrm{Val}(\ell \to p_t,b) \underset{df}{=} \begin{cases} \mathrm{Val}(p_t,b) & \text{if } \mathrm{Val}(\ell,b) = \textit{true} \\ \emptyset & \textit{false} \end{cases}$$

$$\mathrm{Val}(<\ell,p>,b) \underset{df}{=} \{ \prod_{k=o}^{M} b_k \;/\; b_o = e \wedge b_k \in \mathrm{Val}(p, b \cdot \prod_{i=o}^{k=1} b_i) \wedge M = \mathrm{Min}(m \;/\; \mathrm{Val}(\ell, b \cdot \prod_{i=o}^{m} b_i) = \textit{false}\}$$

$$\mathrm{Val}((p_1,\ldots,p_n),b) \underset{df}{=} \bigcup_{i=1}^{n} \mathrm{Val}(p_i,b)$$

$$\mathrm{Val}(\lceil p_1,\ldots,p_n \rceil,b) \underset{df}{=} \bigcup_{\omega\in\Theta_n} \prod_{\substack{i=1 \\ \mathrm{Val}(P_{\omega(i)}.b)\neq\emptyset}}^{n} \mathrm{Val}(p_{\omega(i)},b) \qquad 2)$$

$$\mathrm{Val}(v\lceil p_1,p_2 \rceil,b) \underset{df}{=} \mathrm{Val}(p_2, \mathrm{Int}\ b(v)\ (\mathrm{Int}\ p_1\ (b)))$$

with $\mathrm{Int}\ p\ (b) \underset{df}{=} b \cdot \mathrm{Val}(p,b)$ and $b(v) \in \underline{P}$

$$\mathrm{Val}(p;p_t,b) \underset{df}{=} \bigcup_{b'\in\mathrm{Val}(p,b)} b' \cdot \mathrm{Val}(p_t, b \cdot b')$$

We denote, that the function Val for programs declare the rules of execution of the algorithms, which are described by these programs. Therefore is the domain of Val undecidable in general.

In this interpretation the syntactical elements have the following meaning: The element $v::t$ (resp. $v{:}p_t$) characterises a value assignment (resp. a name assignment).

I (resp.Z) represent the dummy statement (resp. a loop). The function Val restricted to programs will be characterized syntactically by /v/. This element corresponds to the operation "contents of" and can be used as goto or call statement.

2) By Θ_n we mean the set of all permutations ω over $1,\ldots,n$.

$\ell \rightarrow p_t$ is meant to be the conditional statement with the if clause ℓ. If ℓ is not valid, than the empty set of valuations will be produced, i.e. it is present a loop. That is different to the usual interpretations of this element. The element $<\ell,p>$ represents the while statement. The program p will be executed performed repeatedly as long as the condition ℓ is valid. By $v[p_1,p_2]$ a subroutine with the name v will be called. The input--process p_1 and the output-process p_2 realises the information transfer between program and subroutine. In programming languages these processes are used u.o. for parameter declarations. The valuation of the variables of the main program will be pushed. By that it is possible to describe recursive processes.

$p_1;p_2$ meant the sequential execution of the processes, which will be described by the programs p_1 and p_2.

The element $(p_1,\ldots,p_n)$ introduce nondeterministics in such way, that any one of the programs $p_1,\ldots,p_n$ can be executed. In the result the set of valuations arises, which are produced independent by the programs $p_1,\ldots,p_2$. $[p_1,\ldots,p_n]$ describes a parallel process, where the programs $p_1,\ldots,p_n$ will be executed independent to each other as above, but now in result arises the set of valuations, which are produced by multiple supplement of the values for the programs $p_1,p_2,\ldots$ resp. in some sequence. Through the selection of these sequences the time of execution of the processes will be regarded. The two last mentioned elements make it possibly to describe various kinds of parallel and simultaneous modes in execution of the algorithms. [6].[7].

We shall introduce two more relations for programs. For this we give following definition and make use of the notation Valp (b) $\underset{df}{=}$ Val(p,b):

1° The program p_1 is *contained in* program p_2 *in reference to* V' (abbr. $p_1 \leqslant_{V'} p_2$), if Val $p_1 \leqslant_{V'}$ Val p_2.

2° The programs p_1 and p_2 are *equivalent in reference to* V' (abbr. $p_1 \overset{=}{}_{V'} p_2$), if $p_1 \leqslant_{V'} 2$ and $p_2 \leqslant_{V'} p_1$.

As above can be shown: If $V' \subseteq V''$, then from $p_1 \leqslant_{V''} p_2$ (resp. $p_1 \overset{=}{}_{V''} p_2$) implies $p_1 \leqslant_{V'} p_2$ (resp. $p_1 \overset{=}{}_{V'} p_2$).

Example: We give several programs for the function $y = x!$ in the domain of the integers. The symbols $<$, $=$, $+$, $-$, $\cdot$, 0_n, 1_n mean less than, equal, addition, subtraction, product, number 0, number 1 resp..

Program with while element:

$$i::0_n;y::1_n;<i < x, i::i+1_n;y::i\cdot y>$$

Program with goto's:

$$i::0_n;y::1_n;v:(i < x \rightarrow (i::i+1_n;y::i\cdot y;/v/), \tilde{}\, i < x \rightarrow I);/v/$$

Program as a recursive procedure:

$$v:(x = 0_n \rightarrow y::1_n, \tilde{}\, x = 0_n \rightarrow v[x::x-1_n, y::(x+1_n)\cdot y]);/v/$$

In the above stated interpretation all programs produces the same values by every valuation, i.e. they are equivalent among each other.

4. Now we are going to give a connection between programs without subroutines on the one hand and graphs schemes on the other hand.

Let be $S = (N,E,n_i,n_o)$ a graph scheme over $\underline{P}'$, where

- N,E nonempty, finite sets (set of the nodes and set of the edges) and $N = \{n_1,\ldots,n_m\}$.
- $E \subseteq N \times \underline{P}' \times N$ with $e = (e^1,e^2,e^3) \in E$ and $\underline{P}'$ is the set of the programs without subroutines.
- $n_i,n_o \in N$ (n_i is the start of S and n_o is the end of S) and n_o does not possess any successor, i.e. $\{e \,/\, e \in E \wedge e^1 = n_o\} = \emptyset$.
- Each node $n \in N$ is on a path from n_i to n_o. [1]

[1] By a path w from n to n' we understand a sequences of edges $e_1,\ldots,e_l$ with $e_1^1 = n$, $e_l^3 = n'$ and for every $1 \leq k < l$ is valid $e_{k+1}^1 = e_k^3 \cdot 1$ denote the length $l(w)$ of the path w.

To each path $w = (e_1,\ldots,e_l)$ correspond one to one a program $p(w) = e_1^2; \ldots ;e_l^2$. By W_S we mean the set of all pathes from n_i to n_o. For each valuation $b \in W^V$ the execution of S can be described by the function Val S:

$$\text{Val } S\ (b) \underset{df}{=} \bigcup_{w \in W_S} \text{Val}(p(w),b)$$

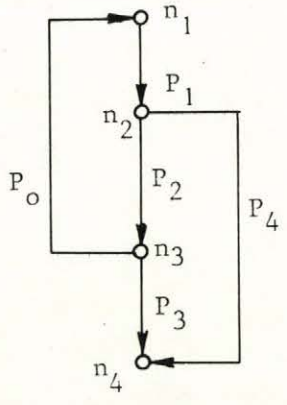
Val S is a function from F. Now we can be proved, that to each graph scheme S over $\underline{P'}$ exists a program $p_S \in \underline{P'}$, with

$$\text{Val } S \underset{V_S}{=} \text{Val } p_s$$

($V_S \subseteq V$ means the set of all variables from programs in E.)
Such a program p_S can be received, that to each node n from N we add one to one a variable from $V-V_s$ with the same symbol and a program term $p_t(n)$ from $\underline{PT}$.

$$p_t(n) \underset{df}{=} \begin{cases} n{:}I & \text{if } n = n_o \\ n{:}(e^2,/e^3/\ )e^1 = n & \text{otherwise} \end{cases}$$

The program p_S can be formed to: $p_S \underset{df}{=} p_t(n_1); \ldots ;p_t(n_m);/n_i/$
As an example we consider the graph scheme in *figure 1*, which correspond to the program



$n_1{:}(p_1;/n_2/);n_2{:}(p_2;/n_3/,p_4;/n_4/);$
$n_3{:}(p_o;/n_1/,p_3;/n_4/);n_4{:}I;/n_1/$.

Figure 1

5. A simple example shall demonstrate the efficiency of the language. In this connection we consider the control for the filling of two tanks with drain (see figure 2).

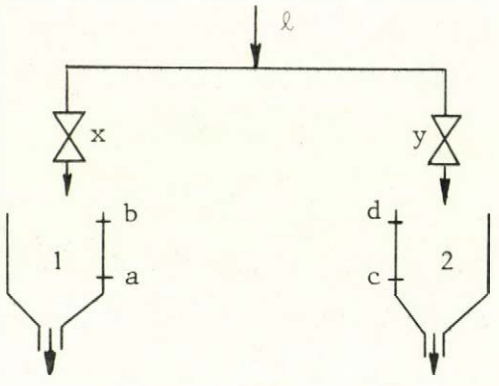


Figure 2

The tank 1 resp. 2 will be filled by the valve x resp.y and the signals a,b resp, c,d gives the level of the filling.

We consider x,y,a,b,c,d as logical variables, so that for ex. the value of x is *true*, if the valve x is opened. The filling of the tanks begins, if the level goes under a resp. c(i.e. the value of a resp. c is *false*). The filling stops, if the level goes over b resp. d(i.e. the value of b resp. d is *true*). The tank 2 shall only then be filled, if the level in tank 1 goes over a (i.e. the value of a is *true*).The whole process is being checked by a logical condition 1 from outside.

Now we consider two variants of the control.

S1: The tanks 1 and 2 will be filled successively.

S2: The tanks 1 and 2 will be filled simultaneously, however the filling of tank 2 will be stopped, if the level in the tank 1 goes under a.

The control of the valves x resp. y will be considered as subroutines v_x resp. v_y. The program p (not specified here) may realise the signals a,b,c,d and the condition ℓ.

During the design of S1 and S2 the subroutines v_x and v_y must not yet been specified.

We obtain for S1 resp. S2 the following programs:

S1: $<\ell,p;/v_x/;/v_y/$

S2: $[<\ell,p;/v_x/>,<\ell,p;/v_y/>]$.

The programs for the control of v_x resp. v_y becomes to:

$$v_x: (\bar{\bar{x}}a \to x::1, b \to x::0, (\bar{x}a \vee \bar{x}b) \to I)$$

$$v_y: (a\bar{\bar{c}}y \to y::1, (\bar{a} \vee d) \to y::0, a(c\bar{y} \vee \bar{d}y) \to I).$$

REFERENCES

1 de Bakker,J.W.; de Roever,W.P.: A calculus for recursive program schemes, Proc.IRIA Symp. on automata, formal languages and programming, North-Holland Amsterdam, 1972, 167-196

2 Dahl,O.J.; Dijkstra,E.W.; Hoare,C.A.R.: Structured programming, Academic Press, New York, 1972

3 Gerber,S.; Haubold,K.: Syntaktisch bewertete Graphschemata mit binärarithmetisch spezialisierter Semantik, Mitt. a.d.Sekt. Math.d.KMU Leipzig,1975

4 Greibach,S.A.: Theory of program structures, schemes,semantics, verification, Lect.Notes in Comp. Science 36, 1975

5 Manna,Z.: Mathematical Theory of Computation, McGraw-Hill Computer Science Series, New York, 1974

6 Mazurkiewicz,A.: Parallel recursive program schemes, Lect.Notes in Comp.Science 32,1975, 75-87

7 Podlovchenko,R.I.: Non-determined algorithm schemata or R-schemata, Lect.Notes in Comp.Science 5,1974, 86-110

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1978

З.Б. МИАДОВИЧ

ГРУППА АВТОМОРФИЗМОВ ПЕРИОДИЧЕСКОЙ СУММЫ КОНЕЧНЫХ АВТОМАТОВ

Институт Автоматики
Технический Университет, Познань

ВВЕДЕНИЕ.

Понятие "*периодический автомат*" как специальный вид "*автомата с переменной структурой*" введено Гиллом в 1963 году [3]. Понятие "*периодическая сумма конечных автоматов*" впервые было указано Гжимала-Буссе в работе [5]. Периодическая сумма автоматов интересна с технической точки зрения, а именно, существует возможность экономичной технической реализации периодической суммы конечных автоматов потому, что существует декомпозиция периодической суммы автоматов - такая, что один из компонентов декомпозиции это просто автономный автомат, а второй - обычный автомат.

1. ПОСТАНОВКА ПРОБЛЕМЫ.

В этой работе будут рассмотрены две проблемы, связанные с группой автоморфизмов закрепленного аналога периодической суммы конечных автоматов, а именно: когда автоморфизмы закрепленного аналога периодической суммы конечных автоматов $A^0, A^1, \ldots, A^{p-1}$ сужены на множества $S^0, S^1, \ldots, S^{p-1}$ автоматов $A^0, A^1, \ldots, A^{p-1}$ являются элементами $\Gamma(A^0), \Gamma(A^1), \ldots, \Gamma(A^{p-1})$ соответственно, и когда эти автоморфизмы будут элементами множеств $ISO(A^0 \to A^1)$, $ISO(A^1 \to A^2), \ldots,$ $ISO(A^{p-1} \to A^0)$.

В работе представлены необходимые и достаточные условия для решения этих задач.

Проблемы, связанные с группами автоморфизмов периодических автоматов, рассмотрены также в работах [1,4,5,9].

1.1. Основные понятия и определения.

В этой работе *автоматом* будем называть тройку (S,Σ,M), где S - конечное, непустое множество состояний; Σ - конечное, непустое множество входов; $M:S\times\Sigma \to S$ - функция следующего состояния.

Автомат есть *сильно связный* тогда и только тогда, если для всяких пар (s,s') состояний A существует $\upsilon\in\Sigma^*$ такое, что $M(s,\upsilon)=s'$, где Σ^* есть множество всех конечных последовательностей элементов Σ.

Пусть p будет положительным числом. *Точно периодический автомат*, далее - просто *периодический автомат*, V есть тройка (S^+,Σ,M^+), где S^+ есть последовательность $S^+_o, S^+_1, \ldots, S^+_{p-1}$ конечных непустых множеств состояний; Σ есть конечное непустое множество входов; M^+ есть последовательность $M^+_o, M^+_1, \ldots, M^+_{p-1}$ функций следующего состояния, где:

$$M^+_a : S^+_a \times \Sigma^+ \to S^+_{a+1 \,(\mathrm{mod}\ p)} \qquad \text{и} \qquad a = 0,1,\ldots,p-1$$

Число p будем называть периодом V.

Закрепленным аналогом V^* *периодического автомата* $V = (S^+, \Sigma, M^+)$ является автомат (S^*,Σ,M^*), где $S^* = S^+_0 \cup S^+_1 \cup \ldots \cup S^+_{p-1}$ и $M^*: S^* \times \Sigma \to S^*$ есть функция переходов, определенная для всяких $s \in S^+_a$, $\sigma \in \Sigma$ $a = 0,1,\ldots,p-1$, следующим образом:

$$M^*(s,\sigma) = M^+_a(s,\sigma).$$

Пусть $A^0 = (S^0,\Sigma, M^0)$, $A^1=(S^1,\Sigma,M^1),\ldots,\ldots,$ $A^{p-1} = (S^{p-1},\Sigma,M^{p-1})$ будут автоматами, пусть $\psi_0: S^0 \to S^1$, $\Psi_1: S^1 \to S^2$, $\Psi_{p-1}: S^{p-1} \to S^0$ будут взаимно однозначными функциями.

Периодическая сумма автоматов $A^0, A^1, \ldots, A^{p-1}$, *связанная с функциями* $\Psi_0, \Psi_1, \ldots, \Psi_{p-1}$ есть периодический автомат (S^+,Σ,M^+) с периодом p,, где S^+ - последовательность $S^0, S^1, \ldots, S^{p-1}$, , а M^+ - последовательность M^+_0, $M^+_1, \ldots, M^+_{p-1}$, , где для всех $s \in S^a$, $\sigma \in \Sigma$, при $a = 0,1,\ldots,p-1$ имеем

$$M^+_a(s,\sigma) = \Psi_a M^a(s,\sigma).$$

Пусть $A = (S,\Sigma,M)$ и $B = (T,\Sigma,N)$ будут автоматами. Функция $\phi: S \to T$ называется *гомоморфизмом* из A в B тогда, когда для всех $s \in S$, $\sigma \in \Sigma$

$$\phi(M(s,\sigma)) = N(\phi(s),\sigma)$$

Функция ϕ называется *изоморфизмом*, если ϕ взаимно однозначна и отображает множество S на множество T. Φ называется *автоморфизмом* A, если Φ - изоморфизм из A в A. Множество всех автоморфизмов A есть группа автоморфизмов, обозначенная через $\Gamma(A)$. Множество всех изоморфизмов из A в B обозначается через $(A \to B)$.

2. РЕЗУЛЬТАТЫ.

Теорема 1.

Пусть $A^0 = (S^0,\Sigma,M^0)$, $A^1 = (S^1,\Sigma,M^1),\ldots,$ $A^{p-1} = (S^{p-1},\Sigma,M^{p-1})$ будут сильно связными автоматами; пусть $\Psi_0: S^0 \to S^1$, $\Psi_1: S^1 \to S^2,\ldots,$ $\Psi_{p-1}: S^{p-1} \to S^0$ будут взаимно однозначными функциями. Пусть $V = (S^+,\Sigma,M^+)$ будет периодической суммой автоматов $A^0,A^1,\ldots,A^{p-1}$, связанной с функциями Ψ_0, $\Psi_1,\ldots,$ Ψ_{p-1}, и $V^* = (S^*,\Sigma,M^*)$ будет закрепленным аналогом V. Пусть для $a = 0,1,\ldots,p-1$ $\alpha_a: S^a \to S^a$ будет функцией $\alpha: S^* \to S^*$, суженной на множество S^a, соответственно, такая, что $\alpha_a \in \Gamma(A^a)$. Тогда существуют:

$\alpha_M \notin \Gamma(A^M)$, $\alpha_{M-1} \in \Gamma(A^{M-1})$ и $\alpha_{M-1} = \Psi_{M-2} \cdots \Psi_{M+1}\ \Psi_M\ \Psi_{M-1}$,

$\alpha_{M-1} = \Psi_M\ \Psi_{M-1} \cdots \Psi_{M+2}\ \Psi_{M-1}$, где $M \in \{0,1,\ldots,p-1\}$ тогда

$\alpha \notin \Gamma(V^*)$..

Доказательство.

Пусть $\alpha \in \Gamma(V^*)$, тогда для всех $s \in S^*$, $\sigma \in \Sigma$ имеем

Это значит, что для всех $s \in S^{M-1}$, $\sigma \in \Sigma$

$$\alpha_M \Psi_{M-1} M^{M-1}(s,\sigma) = \Psi_{M-1} M^{M-1}(\alpha_{M-1}(s),\sigma)$$

и для всех $s \in S^M$, $\sigma \in \Sigma$

$$\alpha_{M+1} \Psi_M M^M(s,\sigma) = \Psi_M M^M(\alpha_M(s),\sigma).$$

Из предположения имеем для всех $s \in S^{M-1}$, $\sigma \in \Sigma$

$$\alpha_{M-1} M^{M-1}(s,\sigma) = M^{M-1}(\alpha_{M-1}(s),\sigma)$$

а также существует $s \in S^M$, $\sigma \in \Sigma$ такое, что

$$\alpha_M M^M(s,\sigma) \neq M^M(\alpha_M(s),\sigma).$$

Из этих условий получим, что

$$\alpha_M \Psi_{M-1} = \Psi_{M-1} \alpha_{M-1}$$

и далее

$$\alpha_M = \Psi_{M-1} \Psi_{M-2} \cdots \Psi_{M+1} \Psi_M.$$

С другой стороны, существует $s \in S^M$, $\sigma \in \Sigma$ такое, что

$$\alpha_{M+1} \Psi_M \neq \Psi_M \alpha_M$$

потому, что $\alpha_M \notin \Gamma(A^M)$..

Когда вместо α_{M+1} в последнее условие можно записать

$$\alpha_{M+1} = \Psi_M \Psi_{M-1} \cdots \Psi_{M+2} \Psi_{M+1}$$

то тогда получим, что

$$\alpha_M \neq \Psi_{M-1} \Psi_{M-2} \cdots \Psi_{M+1} \Psi_M$$

то-есть мы получили противоречие, что доказывает теорему.

Теорема 2.

Пусть $A^0 = (S^0,\Sigma,M^0)$, $A^1 = (S^1,\Sigma,M^1)$,..., $A^{p-1}=(S^{p-1},\Sigma,M^{p-1})$

будут сильно связными автоматами; пусть $\Psi_0: S^0 \to S^1$, $\Psi_1: S^1 \to S^2, \ldots,$ $\Psi_{p-1}: S^{p-1} \to S^0$ будут взаимно однозначными функциями. Пусть $V = (S^+, \Sigma, M^+)$ будет периодической суммой автоматов $A^0, A^1, \ldots A^{p-1}$, связанной с функциями $\Psi_0, \Psi_1, \ldots, \Psi_{p-1}$; пусть $V^* = (S^*, \Sigma, M^*)$ будет закрепленным аналогом V. Пусть $\phi: S^* \to S^*$ будет такой взаимно однозначной функцией, что ее сужения на множества состояний $S^0, S^1, \ldots, S^{p-1}$ будут соответственно равны последовательностям функций $(\Psi_{p-1}\ \Psi_{p-2} \cdots \Psi_1\ \Psi_0)$, $(\Psi_0\ \Psi_{p-1} \cdots \Psi_2\ \Psi_1), \ldots, (\Psi_{p-2}\ \Psi_{p-3} \cdots \Psi_0\ \Psi_{p-1})$. Не существует последовательности $(\Psi_{M-1}\ \Psi_{M-2} \cdots \Psi_{M+1}\ \Psi_M) \notin \Gamma(A^M)$, тогда и только тогда, если $\phi \notin \Gamma(V^*)$, где $M = 0,1,\ldots,p-1$.

Доказательство.

Необходимое условие.

Пусть $\phi \in \Gamma(V^*)$. . Это значит, что для всех

$$\phi(M^*(s,\sigma)) = M^*(\phi(s),\sigma)$$

и далее, - для всех $s \in S^M$, $\sigma \in \Sigma$

$$\phi\ \Psi_M\ M^M(s,\sigma) = \Psi_M\ M^M(\phi(s),\sigma),$$

где $M = 0,1,\ldots,p-1$.

Из предположения имеем для всех $s \in S^M$, $\sigma \in \Sigma$

$$(\Psi_M\ \Psi_{M-1} \cdots \Psi_{M+2}\ \Psi_{M+1})\ \Psi_M\ M^M(s,\sigma) = \Psi_M\ M^M(\Psi_{M-1} \ldots \Psi_M(s),\sigma)$$

С другой стороны, существует $s \in S^M$, $\sigma \in \Sigma$ такое, что

$$(\Psi_{M-1}\ \Psi_{M-2} \cdots \Psi_{M+1}\ \Psi_M)\ M^M(s,\sigma) \neq M^M(\Psi_{M-1}\ \Psi_{M-2} \cdots \Psi_M(s),\sigma)$$

и последовательность функций

$$(\Psi_{M-1}\ \Psi_{M-2} \cdots \Psi_{M+1}\ \Psi_M) \notin \Gamma(A^M).$$

Из двух последних условий мы получили противоречие.

Достаточное условие.

Когда $\phi \notin \Gamma(V^*)$, тогда существует $s \in S^*$ $\sigma \in \Sigma$ такое, что

$$\phi(M^*(s,\sigma)) \neq M^*(\phi(s),\sigma).$$

Пусть $s \in S^M$, где $M \in \{0,1,\ldots,p-1\}$ тогда как вытекает из определения закрепленного аналога периодической суммы автоматов и сужений функции ϕ на множества S^M, S^{M+1} получим, что

$$(\Psi_M \Psi_{M-1} \cdots \Psi_{M+1})\ \Psi_M M^M(s,\sigma) \neq \Psi_M M^M(\Psi_{M-1} \cdots \Psi_M(s),\sigma).$$

Пусть $(\Psi_{M-1} \Psi_{M-2} \cdots \Psi_{M+1} \Psi_M) \in \Gamma(A^M)$, это значит, что для всех $s \in S^M$, $\delta \in \Sigma$

$$(\Psi_{M-1}\Psi_{M-2} \cdots \Psi_{M+1} \Psi_M)\ M^M(s,\sigma) = M^M(\Psi_{M-1} \Psi_{M-2} \cdots \Psi_M(s),\sigma).$$

В этих двух условиях мы получили противоречие.

Теорема 3.

Пусть $A^0 = (S^0,\Sigma,M^0)$, $A^1 = (S^1,\Sigma,M^1),\ldots,A^{p-1} = (S^{p-1},\Sigma, M^{p-1})$ будут сильно связными изоморфными автоматами; пусть $\Psi_0:S^0 \to S^1$, $\Psi_1:S^1 \to S^2,\ldots,\Psi_{p-1}:S^{p-1} \to S^0$ будут взаимно однозначными функциями, пусть $\alpha_0:S^0 \to S^0$, $\alpha_1:S^1 \to S^1$, $\ldots$, $\alpha_{p-1}:S^{p-1} \to S^{p-1}$ будут взаимно однозначными функциями и пусть $V = (S^+,\Sigma,M^+)$ будет периодической суммой автоматов $A^0,A^1,\ldots,A^{p-1}$, связанной с функциями $\Psi_0, \Psi_1,\ldots, \Psi_{p-1}$.

Пусть $\phi:S^* \to S^*$ будет такой взаимно однозначной функцией, что ее сужения на множества $S^0,S^1,\ldots,S^{p-1}$ будут соответственно равны $\alpha_0, \alpha_1,\ldots, \alpha_{p-1}$.

$\Psi_0, \Psi_1,\ldots, \Psi_{p-1}$ являются изоморфизмами тогда и только тогда, если не существует $\phi \in \Gamma(V^*)$ такая, что $\alpha_{M-1} \in \Gamma(A^{M-1})$ и $\alpha_M \notin \Gamma(A^M)$, где $M \in \{0,1,\ldots,p-1\}$.

Доказательство.

Пусть $\phi \in \Gamma(V^*)$; при этом мы получим, что для всех $s \in S^*$, $\sigma \in \Sigma$

$$\phi\, M^*(s,\sigma) = M^*(\phi(s),\sigma).$$

Это также означает, что для всех $s \in S^{M-1}$, $\sigma \in \Sigma$ мы имеем

$$\phi\, \Psi_{M-1}\, M^{M-1}(s,\sigma) = \Psi_{M-1}\, M^{M-1}(\phi(s),\sigma)$$

и далее, как вытекает из сужения функции ϕ на множества S^{M-1} и S^M, :

$$\alpha_M\, \Psi_{M-1}\, M^{M-1}(s,\sigma) = \Psi_{M-1}\, M^{M-1}(\alpha_{M-1}(s),\sigma).$$

Из предположения, что $\alpha_{M-1} \in \Gamma(A^{M-1})$ имеем, что для всех $s \in S^{M-1}$, $\sigma \in \Sigma$

$$\alpha_{M-1}\, M^{M-1}(s,\sigma) = M^{M-1}(\alpha_{M-1}(s),\sigma).$$

Из этих условий получаем уравнение

$$\alpha_M\, \Psi_{M-1} = \Psi_{M-1}\, \alpha_{M-1}.$$

Необходимое условие.

Из последнего уравнения получим для всех $s \in S^M$, $\sigma \in \Sigma$ выражение:

$$\alpha_M = \Psi_{M-1}\, \alpha_{M-1}\, \Psi_{M-1}^{-1}.$$

С другой стороны, из преположения, что $\alpha_M \notin \Gamma(A^M)$, имеем, что существует $s \in S^M$, $\sigma \in \Sigma$ такое, что

$$\alpha_M\, M^M(s,\sigma) \neq M^M(\alpha_M(s),\sigma)$$

или

$$\Psi_{M-1}\, \alpha_{M-1}\, \Psi_{M-1}^{-1}\, M^M(s,\sigma) \neq M^M(\Psi_{M-1}\, \alpha_{M-1}\, \Psi_{M-1}^{-1}(s),\sigma).$$

Приведенные выше неравенства будут истинными только тогда, если $\Psi_{M-1} \notin ISO\ (A^{M-1} \to A^{M})$ потому, что $\alpha_{M-1} \in \Gamma(A^{M-1})$. Тогда мы получим противоречие.

Достаточное условие.

Превратим условие

$$\alpha_M\ \Psi_{M-1} = \Psi_{M-1}\ \alpha_{M-1}$$

в вид

$$\Psi_{M-1} = \alpha_M^{-1}\ \Psi_{M-1}\ \alpha_{M-1}.$$

Далее имеем, что для всех $s \in S^{M+1}$, $\sigma \in \Sigma$

$$\Psi_{M-1}\ M^{M-1}(s,\sigma) = M^{M}(\Psi_{M-1}(s),\sigma)$$

потому, что $\Psi_{M-1} \in ISO(A^{M-1} \to A^{M})$ и в последствии получим, что для всех $s \in S^{M-1}$, $\sigma \in \Sigma$

$$\alpha_M^{-1}\ \Psi_{M-1}\ \alpha_{M-1}\ M^{M-1}(s,\sigma) = M^{M}(\alpha_M^{-1}\ \Psi_{M-1}\ \alpha_{M-1}(s),\sigma)$$

и далее

$$\alpha_M^{-1}\ M^{M}(\Psi_{M-1}\ \alpha_{M-1}(s),\sigma) = M^{M}(\alpha_M^{-1}\ \Psi_{M-1}\ \alpha_{M-1}(s),\sigma)$$

из-за предположений $\Psi_{M-1} \in ISO\ (A^{M-1} \to A^{M})$ и $\alpha_{M-1} \in \Gamma(A^{M-1})$.

Это значит, что мы получим противоречие, так как $\alpha_M \in \Gamma(A^{M})$.

Л И Т Е Р А Т У Р А

[1] B.H.Barnes: On the automorphism groups of periodic automata., Information and Control, 20 2 1972 125-134.

[2] A.C.Fleck: Isomorphism groups of automata., J.Assoc.Comp.Mach., 9 4 1962 469-476.

[3] A.Gill: Time-varying sequential machines., J.Franklin Institute, 276 1963 519-539

[4] J.Grzymała-Busse: Automorphism of total periodic automata., Bull.Soc.Amis Sci.Let.Poznań Sér.B,22 1970/71 113-123.

[5] J.Grzymała-Busse: On the connectivity of the periodic sum of automata Math.Found. of Comp.Sci.,High Tatras, Sep 3-8, 1973 231-234.

[6] J.Grzymała-Busse: On the automorphisms of periodic automata extensions Bull de L'Academic Polonaise des Sci. Sér des Sci math., ast. et phys. Vol XXII, No 3 1974 325-331.

[7] Z.Miądowicz, B.Mikołajczak: On the connectivity of the periodic sum of finite atuomata. Found. of Contr.Eng. 1, 2, 1977 97-102.

[8] Z. Miądowicz: Some problems concerning with the periodic sum of finite automata. 5-th IFAC-Symp. on Discrete Systems, Vol.5, 94-102, March 14-19 1977.

[9] Z. Miądowicz: The automorphism group of periodic sum of finite automata. Found. of Contr.Eng. 2, 4 1977 195-203.

[10] G.P. Weeg: The structure of an automaton and its operation--perserving transformation group. J.Assoc, Comp.Mach 9 3 1962 345-349.

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1978

L. SZLÁVIK, I. NAGY

НЕКОТОРЫЕ ПРОБЛЕМЫ АВТОМАТИЧЕСКОЙ ЛОКАЛИЗАЦИИ ОШИБОК

Исследовательский Институт Вычислительной
Техники и Автоматизации ВАН

1. ПОСТАНОВКА ПРОБЛЕМЫ

В настоящей статье мы занимаемся автоматической локализацией ошибок смонтированных цифровых печатных плат, основанной на применении проверяющих последовательностей, а также на дополнительных измерениях, проведенных во внутренних точках исследуемой сети.

Изложенная здесь проблема является частью находящейся в стадии разработки проекта "системы проверки плат" [1] в отделе цифровой техники ИИВТиА ВАН.

Основная идея метода локализации ошибок такова же, как "техник" делает при ручной локализации ошибок с помощью тестпрограммы, выполненной автоматическим тестом. То есть, в ходе выполнения тестпрограммы приостановим ее при тесте, обнаруживающем первую ошибку. На определенных деталях сети производятся дополнительные измерения с повторением уже выполненной части тестпрограммы, имея при этом следующие информации:

1/ Обнаруженный признак ошибки на первичных выходах сети;
2/ Выполненные тесты до теста, обнаруживающего первую ошибку;
3/ Логическо-топологические данные сети;
4/ В данной фазе локализации результаты в предшествующем выполненных дополнительных измерений.

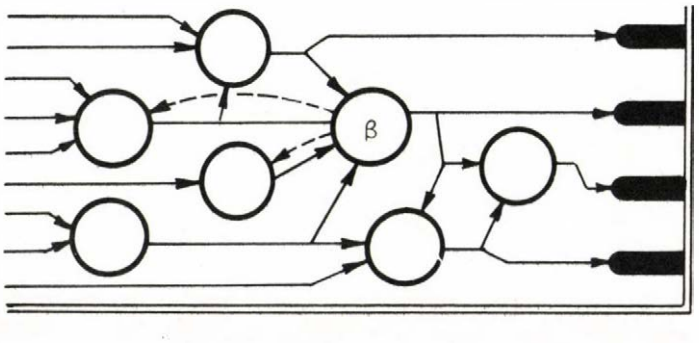
Метод, применяемый здесь, способен локализировать все такие ошибки смонтированных цифровых печатных плат, которые вообще возможно обнаружить с помощью тестпрограммы.

В дальнейшем мы опишем важнейшую часть процедуры локализации ошибок так называемую селективную разметку испытаний, указывающую подряд детали, подлежащие измерению, от выхода детали, к ее входам вплоть до места причины ошибки.

Селективная разметка испытаний означает следующее: первой деталью выбирается для дополнительных измерений, та деталь на выходе которой обнаружена ошибка. Очевидно, что выход этой детали будет первичном выходом исследуемой сети. /Если имеется больше таких деталей, то из них выбираем любую и можно доказать, что независимо от выбора, всегда будет локализирована одна ошибка./

Если деталь β уже выбиралась в какой-то предшествующей фазе разметки испытаний, то повторяется испытателем нужная часть тестовой последовательности. На основании этого испытатель устанавливает - деталь β годна или негодна по отношению к тестовой последовательности. В случае негодной детали, он заменяет ее и возобновляет процесс испытания сети. Если деталь после этого годна, то дальнейшие испытания будут проведены только на тех деталях, подключенных к входам детали β , которые являлись исключительно-определяющимися входами детали .

2. МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ



где: ~ деталь, и

~ селективная разметка испытаний.

ТЭ3

Определение таких существенных входов становилось возможным введением понятия определяющего множества.

Перед тем, как определить вышеуказанное, рассмотрим деталь как автомат.

ОПРЕДЕЛЕНИЕ 1

ДЕТАЛЬ - АВТОМАТ

Пусть, множество входов детали A: $X = \{x_1, x_2, \ldots, x_n\}$ и ее входов K.

Тогда деталь A может быть задана следующим автоматом:

$$A = \langle B_1; B_n; Q; \delta \rangle$$

Где B_1: выходной алфавит /множество значений K/; $B_1 = \{0,1\}$
B_n: входной алфавит /множество значений X/; $B_n = \{0,1\}^n$
Q : множество внутренних состояний; $Q = \{q_1, q_2, \ldots, q_n\}$
δ : функция переходов; $\delta : B_n X Q \to Q X B_1$

После этого рассмотрим определяющее множество.

ОПРЕДЕЛЕНИЕ 2

ОПРЕДЕЛЯЮЩЕЕ МНОЖЕСТВО

В каком-то состоянии q_i деталь-автомат A некоторое

$$D_j^o(q_i) = \{x_{j1}^o, x_{j2}^o, \ldots, x_{jk}^o\} \; ; \; x_{ji}^o \in \{0,1\} \; , \; i \in \{1,2,\ldots,k\}$$

значение множества

$$D_j(q_i) = \{x_{j1}, x_{j2}, \ldots, x_{jk}\} \; ; \; D_j(q_i) \subseteq X$$

называется определяющим множеством, если существует $q_\ell \in Q$ и $K^o \in B_1$, что

$$\forall G_j^o(q_i) [\delta [D_j^o(q_i) \cup G_j^o(q_i); q_i] = (q_\ell; K^o)] \; , \qquad (*)$$

где

$$G_j(q_i) = X \setminus D_j(q_i) \ .$$

ОПРЕДЕЛЕНИЕ 3

ЭКВИВАЛЕНТНОСТЬ ОПРЕДЕЛЯЮЩИХ МНОЖЕСТВ

Определяющее множество $D_j^o(q_i)$ эквивалентно определяющему множеству

$$D^o(q_i) \ ; \quad D_j^o(q_i) \in E[D^o(q_i)] \ ,$$

тогда и только тогда, если

$$\forall G_j^o(q_i) \ G_\ell^o(q_i)[\delta[D_j^o(q_i) \ G_j^o(q_i); q_i] \equiv \delta[D_\ell^o(q_i) \ G^o(q_i); q_i]] \ .$$

ОПРЕДЕЛЕНИЕ 4

ОГРАНИЧЕННАЯ ЭКВИВАЛЕНТНОСТЬ ОПРЕДЕЛЯЮЩИХ МНОЖЕСТВ

Определяющее множество $D_j^o(q_i)$ есть ограниченное эквивалентное множество определяющего множества $D_\ell^o(q_i)$; $D_j^o(q_i) \in R_s[D_\ell(q_i)]$, тогда и только тогда, если

$$D_j^o(q_i) \in E[D_\ell^o(q_i)] \ \text{и} \ D_j^o(q_i) \subseteq D_\ell^o(q_i).$$

ОПРЕДЕЛЕНИЕ 5

САМАЯ ОГРАНИЧЕННАЯ ЭКВИВАЛЕНТНОСТЬ ОПРЕДЕЛЯЮЩИХ МНОЖЕСТВ

Определяющее множество $D_j^o(q_i)$ есть самое ограниченное эквивалентное множество определяющего множества $D_\ell^o(q_i)$;
$D_j^o(q_i) \in R_{LS}[D_\ell^o(q_i)]$, тогда и только тогда, если

$$D_j^o(q_i) \in R_s[D_\ell^o(q_i)] \ \text{и} \ R_s[D_j^o(q_i)] = \{D_j^o(q_i)\} \ .$$

3. ВЫВОДЫ

После этого наши существенные постановления следующие:

3.1./ Любое значение множества $X^{o} = \{x_1^{o}, x_2^{o}, \ldots, x_n^{o}\}$ принадлежащего к полному множеству входов X является определяющим множеством в каждом состоянии деталь-автомата A. Так как $D_j(q_i) \equiv X \Rightarrow G_j(q_i) = \emptyset$, поэтому выполнение условия (*) очевидно.

3.2./ Элементами множества $D_j(q_i)$ принадлежащего какому-нибудь множеству $D_j^{o}(q_i) \in R_{LS}[D_\ell^{o}(q_i) = X^{o}]$ являются те входы, которые однозначно определяют переход деталь-автомата, находящегося в состоянии q_i , но пропуская один из элементов множества $D_j(q_i)$ переход деталь-автомата станет уже непригодным.

3.3./ Пусть множество $R_{LS}[D_\ell^{o}(q_i) = X^{o}]$ содержащее самые ограниченные эквиваленты множества X^{o}, где X^{o} множество входов деталь-автомата A при состоянии q_i - определяющее множество /см. 1./. Можно доказать, что для любого определяющего множества $D_j^{o}(q_i) \in R_{LS}[D_\ell^{o}(q_i) = X^{o}]$ достаточно провести исследование только на деталях подключенных к входам $x_{j1}, x_{j2}, \ldots, x_{jk}$ детали A, где

$$\{x_{j1}, x_{j2}, \ldots, x_{jk}\} = D_j(q_i) \subseteq X .$$

Поэтому целесообразно разметку исследований осуществить по любому элементу с наименьшей мощностью множества

$$R_{LS}[D_\ell^{o}(q_i) = X^{o}] .$$

Таким образом осуществляя разметку исследований в сети от ошибочного выхода к входам мы заведомо найдем ошибочную деталь.

4. ЗАКЛЮЧЕНИЕ

Из этого следует, что о вышесказанном способе локализации ошибок вобщем можно сказать следующее:

4.1./ С его помощью все ошибки локализируемы, покрытые наборами тестов.

4.2./ В процедуре применяемая селективная разметка исследований обеспечит, что не увеличивается пропорционально время выполнения локализации ошибок, а также не увеличивается пропорционально потребляемая емкость памяти вычислительной машины при увеличении размера сети.

ЛИТЕРАТУРА:

1. **Ivics J., Szlávik L.:** Некоторые вопросы проверки смонтированных печатных плат. "Доклады НКС СЭВ 1-15.1, 1977 г., Варшава". /См. в этом же сборнике./

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1978

K. SAPIECHA

О ВЗАИМОЗАВИСИМОСТЯХ МЕЖДУ D-АЛГОРИТМОМ И АЛГОРИТМАМИ, ИСПОЛЬЗУЮЩИЕ ПОНЯТИЕ БУЛЕВСЙ ПРОИЗВОДНОЙ

Институт Вычислительной Техники
ВТУ Варшава

Одним из основных компонентов любой системы автоматического проектирования логических схем является блок генерирования тестов.

Тестом неисправности j/α /постоянное значение α на j-том соединении/ модуля **M** называется такой набор входных переменных $X^{j/\alpha}$, что существует выходной полюс Y_i, состояние которого позволяет отличить исправный модуль от неисправного.

Из этого непосредственно вытекает, что совокупность тестовых наборов $T^{j/\alpha}$ для неисправности j/α можно вычислить используя булевую производную. Под Булевой производной функции $F_i(X)$ по переменной x_j понимается следующая функция:

$$\frac{\partial F_i}{\partial x_j} = F_i(x_1,\ldots, x_j\ldots, x_n) \oplus F_i(x_1,\ldots, x_{j-1}, x_j', x_{j+1},\ldots,x_n)$$

$\oplus$ - сумма по модулю 2,
x_j' - инверсия переменной x_j.

Как видно, второе слагаемое в правой части выражения отличается от первого инверсным значением переменной x_j. Это соответствует изменению значения сигнала x_j в узле j, которое вызвано, например, неисправностью узла j. Заметим, что если $H_{ij}=\frac{\partial F_i}{\partial x_j}$, то это обозначаем, как

$$(H_{ij}(X^{j/\alpha})=1) \Leftrightarrow F_i(X_1^{j/\alpha},\ldots,1,\ldots,X_n^{j/\alpha}) \neq F_i(X_1^{j/\alpha},\ldots,0,\ldots,X_n^{j/\alpha})$$

$\uparrow$ ——— x_j ——— $\uparrow$

Для внутреннего узла k имеем:

$$F_i = G_i(X, f_k)$$

$$\frac{\partial F_i}{\partial f_k} = G_i(X, f_k) \oplus G_i(X, f_k')$$

и соответственно

$$(H_{ik}(X^{k/\alpha}) = 1) <=> G_i(X^{k/\alpha}, 1) \neq G_i(X^{k/\alpha}, 0),$$

что конечно удовлетворяет условию теста неисправности k/α.

Из выше сказанного следует, что булева производная оказывается подходящей для вычисления тестов. Но основной трудностью применения булевой производной в системах автоматического вычисления тестов, которые были разработаны для практических исследований, является очень сложная процедура вычисления функции $\frac{\partial F_i}{\partial f_k}$.

С целью упрощения этой процедуры, многие из авторов занимались структурным подходом для вычисления булевой производной. Легко видеть, например, для элемента И /AND/ получаем:

$$F(X) = x_1 x_2 \ldots x_n$$

$$\frac{\partial F}{\partial x_j} = x_1 x_2 \ldots x_{j-1} x_{j+1} \ldots x_n \cdot$$

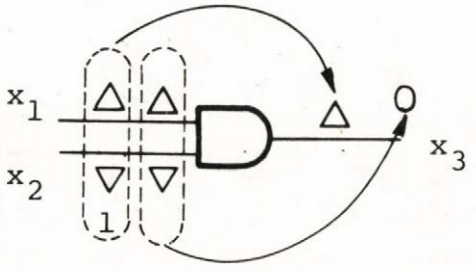
Это выражение очень простое, затем легко вычислить функцию, которая описывается этим выражением. Соответствующее выражение можно получить для других логических элементов. Таким образом, предлагается расчитывать булеву производную элемент за элементом, начиная от выходов схемы. Однако нетрудно заметить, что этот метод допустим только тогда, когда в сети нет разветвлений. Очень сложным является тоже его расширение в случае кратных неисправностей.

Исходя из этого будет показано, что общий структурный метод вычисления булевой производной заключается в том, чтобы анали-

тически описать процесс параллельного распространения в схеме эффекта ее неисправности /так называемой D-информации/.

Как уже было сказано, булева производная функции F по переменной x_j определяет все выходные наборы схемы, реализующей функцию F, на которых возбуждение переменной x_j вызывает возбуждение выхода схемы. Это возбуждение может быть типа 0→1 или 1→0. Пусть Δ обозначает переход 0→1, а ∇ переход 1→0. Рассмотрим таблицу истинности соответствующую элементу И в выше определенной логике:

И	0	1	Δ	∇
0	0	0	0	0
1	0	1	Δ	∇
Δ	0	Δ	Δ	0
∇	0	∇	0	∇



Условия возбуждения выхода элемента И можно записать используя булевые переменные, определенные как ниже:

$$\hat{x}_i = \begin{cases} 1, & \text{сигнал } x_i \text{ изменяется } 0\to1 \\ 0, & \text{в противном случае} \end{cases}$$

$$\check{x}_i = \begin{cases} 1, & \text{сигнал } x_i \text{ изменяется } 1\to0 \\ 0, & \text{в противном случае} \end{cases}$$

Таким образом мы получаем:

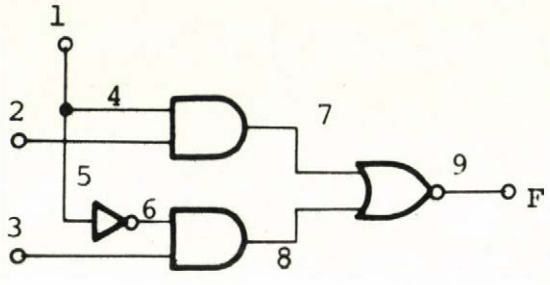
$$\hat{x}_3=\hat{f}^u(x_1,x_2,\hat{x}_1,\hat{x}_2,\check{x}_1,\check{x}_2)=x_1\hat{x}_2+\hat{x}_1x_2+\hat{x}_1\hat{x}_2=(x_1+\hat{x}_1)(x_2+\hat{x}_2)(\hat{x}_1+\hat{x}_2)$$

$$\check{x}_3=\check{f}^u(x_1,x_2,\hat{x}_1,\hat{x}_2,\check{x}_1,\check{x}_2)= x_1\check{x}_2+\check{x}_1x_2+\check{x}_1\check{x}_2=(x_1+\check{x}_1)(x_2+\check{x}_2)(\check{x}_1+\check{x}_2)$$

Условия для остальных типов логических элементов можно получить подобным образом.

Для обобщения выше указанной конструкции, нужно использовать

метод суперпозиции. Так как функция F может быть вычислена с помощью суперпозиции элементарных функций f_ℓ, выполняемых отдельными элементами схемы. Таким образом, функции $\hat{F}$ и $\check{F}$ можно вычислить при помощи суперпозиций элементарных функций $\hat{f}^\alpha$ и $\check{f}^\alpha$, $\alpha \in \{$И, И-НЕ, ИЛИ, ИЛИ-НЕ$\}$. Чтобы определить необходимые для возбуждения условия выхода схемы, надо установить входное возбуждения, а затем вычислить функции $\hat{F}$ и $\check{F}$. Например, для схемы показанной на рисунке, функции $\hat{F}$ и $\check{F}$ представляются как видно.



$$\hat{x}_4=\hat{x}_5=\hat{x}_1$$
$$\check{x}_4=\check{x}_5=\check{x}_1$$
$$\hat{x}_6=\check{x}_5$$
$$\check{x}_6=\hat{x}_5$$
$$\hat{x}_7=(x_4+\hat{x}_4)(x_2+\hat{x}_2)(x_4+\hat{x}_2)$$
$$\check{x}_7=(x_4+\check{x}_4)(x_2+\check{x}_2)(\check{x}_4+\check{x}_2)$$
$$\hat{x}_8=(x_3+\hat{x}_3)(x_6+\hat{x}_6)(\hat{x}_3+\hat{x}_6)$$
$$\check{x}_8=(x_3+\check{x}_3)(x_6+\check{x}_6)(\check{x}_3+\check{x}_6)$$
$$\hat{F}=\hat{x}_9=(x'_7+\check{x}_7)(x'_8+\check{x}_8)(\check{x}_7+\check{x}_8)$$
$$\check{F}=\check{x}_9=(x'_7+\hat{x}_7)(x'_8+\hat{x}_8)(\hat{x}_7+\hat{x}_8)\ .$$

Установление входного возбуждения $(\hat{x}_1,\hat{x}_2,\hat{x}_3)=(0,0,1)$ и $(\check{x}_1,\check{x}_2,\check{x}_3)=(0,1,0)$ приводит к результатам
$\hat{F}(x_1,x_2,x_3,0,0,1,0,1,0)= x_1$ и $\check{F}(x_1,x_2,x_3,0,0,1,0,1,0)=x'_1$

Это обозначает, что возбуждение 1→0 переменной x_2, и 0→1 переменной x_3 вызовет возбуждение 0→1 выхода схемы тогда, когда $x_1=1$, либо возбуждение 1→0, когда $x_1=0$.

Нетрудно заметить, что между функциями $\hat{F}$, $\check{F}$ и $\frac{\partial F}{\partial x_j}$ должна существовать какая то связь. Можно доказать см. в[1] /доказательство здесь обойдем/, что

$$\frac{\partial F}{\partial x_j} = \hat{F}(X, \hat{C}_j, \check{C}_j) + F(X, \hat{C}_j, \check{C}_j)$$

Когда j не разветвляется, или

$$\frac{\partial F}{\partial x_j} = \bar{\hat{F}}(X, \hat{C}_j, \check{C}_j) + \bar{\check{F}}(X, \hat{C}_j, \check{C}_j) \ .$$

Когда j является символом разветвления. Функции $\bar{\hat{F}}$ и $\bar{\check{F}}$ получаются из функции $\hat{F}$ и $\check{F}$ путем простой коррекции [1]. Вектор $(X, \hat{C}_j, \check{C}_j)$ соответствует возбуждению соединения j /необязательно входной переменной/.

Эти зависимости показывают, что структурный метод вычисления булевой производной заключается в том, чтобы аналитически описать вышеуказанный четырехзначный модуль.

Теперь заметим, что алгебры $<\{0,1,\Delta,\nabla\}, \hat{f}^{\alpha}, \check{f}^{\alpha}>$ и $<\{0,1,\bar{D},D\}, \tilde{h}^{\alpha}>$ являются изоморфными. Например, для элемента И выполняется:

И	0	1	Δ	∇
0	0	0	0	0
1	0	1	Δ	∇
Δ	0	Δ	Δ	0
∇	0	∇	0	∇

И	0	1	$\bar{D}$	D
0	0	0	0	0
1	0	1	$\bar{D}$	D
$\bar{D}$	0	$\bar{D}$	$\bar{D}$	0
D	0	D	0	D

Следовательно, структурный метод вычисления булевой производной приводит нас к аналитическому представлению D-алгоритма. Таким образом, они эквивалентны и выбор одного из двух допустимых вариантов вычисления зависит от потребности.

Л И Т Е Р А Т У Р А

1 **K. Sapiecha:** Алгебраические модели переключающих сетей и их применение в области анализа и диагностики логических схем. **Prace Naukowe PW, Elektronika Nr. 35.**

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1978

АНДРЕЙ КРАСЬНЕВСКИ

РАЗМЫТЫЙ АВТОМАТ В СИСТЕМЕ ДИНАМИЧЕСКОГО УПРАВЛЕНИЯ РАСПРЕДЕЛЕНИЕМ ИНФОРМАЦИИ В СЕТИ СВЯЗИ

Политехнический Институт, Варшава

ВВЕДЕНИЕ

Децентрализованная система динамического управления автоматически коммутируемой сетью связи должна обладать особенностью адаптации к изменяющимся условиям, т.е. для каждой ситуации на сети связи система управления должна выбрать оптимальный план распределения потоков информации. План распределения потоков информации определяет метод выбора транзитных узлов в процессе составления соединения между вызывающим и вызываемым узлом.

В работах [1], [2] и других предложен метод управления, в котором коммутируемая сеть связи с изменяющимися во времени параметрами (потоки передаваемой информации, тяготения, ёмкости ветвей, возможность выхода из строя ветвей и узлов коммутации) рассматривается в качестве случайной среды, а система управления - в качестве коллектива вероятностных автоматов взаимодействующих с этой средой. Задачу оптимального управления распределением информации в сети связи можно тогда решать методами теории оптимального поведения автомата в случайной среде.

1. ВЕРОЯТНОСТНЫЙ АВТОМАТ В СЛУЧАЙНОЙ СРЕДЕ

Определим формально понятие вероятностного автомата взаимодействующего со случайной средой.

Случайная среда с дискретным временем $E(к)$ определяется как

$$E(к) = \langle У, X, C(к) \rangle$$

где: $У = \{ y_1, y_2, \ldots, y_n \}$ – множество допустимых входов среды,

$X = \{0,1\}$ – множество входов (действий) среды,

$C(к) = \{ c_1(к), c_2(к), \ldots, c_n(к) \}$

$C(к)$ определяет для каждого допустимого входа вероятность появления в момент к действия $x(к) = 1$

$$c_i(к) = P\{ x(к) = 1 \mid y(к) = y_i \}.$$

Вероятностный автомат с переменной структурой $M(к)$ определяется как

$$M(к) = \langle A, X, У, П(к) \rangle$$

где: $A = \{ a_1, a_2, \ldots, a_n \}$ – множество состояний автомата,

$X = \{0,1\}$ – входной алфавит (множество входов автомата),

$У = \{ y_1, y_2, \ldots, y_n \}$ – выходной алфавит (множество выходов – действий автомата),

$П(к) = \{ \pi_1(к), \pi_2(к), \ldots, \pi_n(к) \}$, $\sum_{i=1}^{n} \pi_i(к) = 1$.

$\pi_i(к)$ определяет вероятность появления состояния a_i автомата в момент к

$$\pi_i(к) = P\{ a(к) = a_i \}$$

Будем принимать, что в каждый момент времени состояние автомата определяет его выход (действие), так что и следовательно

$$a(к) = a_i \Leftrightarrow y(к) = y_i,$$

$$\pi_i(к) = P\{ y(к) = y_i \}.$$

Говорят, что автомат $M(к)$ взаимодействует со средой $E(к)$, если в каждый момент времени к действие автомата $y(к)$ является входом среды, а значение входной переменной автомата является выходом среды (рис.1).

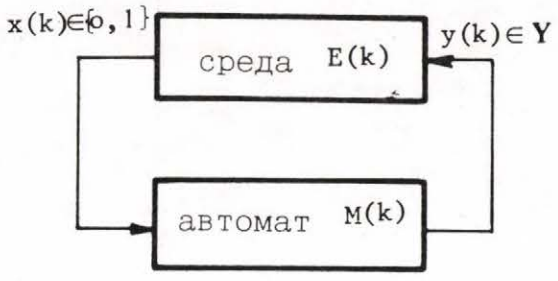


Рис. 1. Взаимодействие автомата со случайной средой

Если действие автомата y(k), произведенное в момент k, влечет за собой действие среды x(k) = 1 говорят, что автомат получил штраф, в противном случае (x(k) = 0) говорят, что автомат получил поощрение. Следовательно

$c_i(k)$ – вероятность штрафа за действие $y(k) = y_i$.

Мерой целесообразности поведения автомата M(k) в среде E(k) является величина математического ожидания среднего числа штрафов за данный период времени

$$Q(K) = E\{\frac{1}{K}\sum_{k=1}^{K} x(k)\}$$

В автомате с переменной структурой в каждый момент времени происходит модификация вероятностей появления состояний (действий) в зависимости от реакции среды. Алгоритм модификации (адаптации) автомата может быть вообще представлен в виде [4]

$$\Pi(k) = T\,(\Pi(k),\ y(k),\ x(k)).$$

Существуют многие алгоритмы адаптации [5]. Большинство из них является алгоритмами с модификацией в случае поощрения и штрафа. Выполняют они следующие условия:

Если $y(k) = y_i$, то

$$\pi_i(k+1) \begin{cases} > \pi(k), & x(k) = 0 \\ < \pi(k), & x(k) = 1 \end{cases}$$

$$\pi_j(k+1) \begin{cases} < \pi(k), & x(k) = 0 \\ > \pi(k), & x(k) = 1 \end{cases}$$

$j \neq i$

Суть этого алгоритма заключается в том, что если автомат в данный момент был поощрен (оштрафован) за действие y_i, вероятность появления этого действия в следующий момент повышается (уменьшается), а всех других действий уменьшается (повышается).

2. Вероятностный автомат как система управления распределением информации в сети связи

Пусть сеть связи имеет N узлов $y_1, y_2, \ldots, y_N$. Рассмотрим процесс составления соединения от узла y_i к узлу y_j. Пусть соединения от y_i к y_j поступающие в дискретные моменты времени $k = 1,2,\ldots,K$ могут устанавливаться по направлениям (транзитным узлам) $y_1^{ij}, y_2^{ij}, \ldots, y_n^{ij}$ (рис. 2).

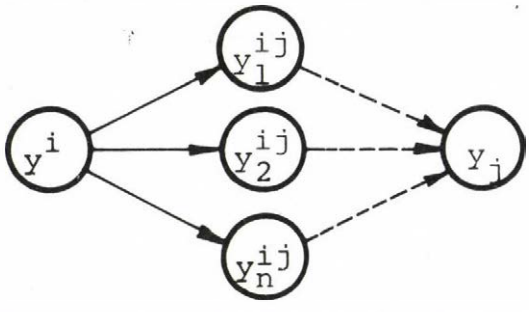


Рис. 2. Процесс составления соединения в сети связи

Вызывающему узлу y_i и узлу назначения y_j поставим в соответствие случайную среду с дискретным временем

$$E^{ij}(k) = \langle Y^{ij}, X^{ij}, C^{ij}(k) \rangle$$

где: $Y^{ij} = \{ y_1^{ij}, y_2^{ij}, \ldots, y_n^{ij} \}$ – множество допустимых транзитных

направлений от y_i к y_j

$c_l^{ij}(к)$ - вероятность отказа при составлении соединения от транзитного узла y_l^{ij} к вызываемому узлу y_j и вероятностный автомат с переменной структурой

$$M^{ij}(к) = \langle A^{ij}, X, Y^{ij}, \Pi^{ij}(к) \rangle$$

где: $п_l^{ij}(к)$ - вероятность выбора транзитного направления y_l^{ij}.

При поступлении вызова от y_i к y_j в момент к выбор транзитного направления осуществляется автоматом $M^{ij}(к)$ (вероятность выбора направления y_l^{ij} равна $п_l^{ij}(к)$). Если в выбранном напрвлении нет свободного канала, выбирается другое направление пока не найдётся направления со свободным каналом. По выбранному направлению производится транзитное проключение соединения. Если выбранный узел является узлом назначения y_j, то установливается соединение с требуемым абонентом узла y_j и автомат M^{ij} получает поощрение. Если транзитный узел не является узлом назначения, весь процесс выбора следующего узла повторяется. Если в узле не нашлось ни одного направления со свободным каналом или число транзитов обслуживаемого вызова превысило допустимое, вызов получает отказ и автомат M^{ij} будет оштрафован.

Вероятностный автомат может осуществлять тоже детерминированный выбор транзитного направления - выбирается направление соответствующее максимальному элементу $\Pi^{ij}(к)$ [2], [3].

Для предложенного метода управления распределением потоков информации в узле коммутации можно доказать, что задача минимизации числа отказов для потока вызовов от узла y_i к узлу y_j соответствует задачи оптимального поведения автомата $M^{ij}(к)$ в среде $E^{ij}(к)$, т.е. задачи минимизации величины математического ожидания среднего числа штрафов за данный период времени

$$Q^{ij}(К) = E\left\{\frac{1}{k}\sum_{k=1}^{k} x^{ij}(К)\right\}.$$

Величина математического ожидания среднего числа штрафов получаемых автоматом $M^{ij}(к)$ для данной среды $E^{ij}(к)$ зависит от алгоритма модификации автомата. В работе [2] исследован алгоритм адаптации автомата, который обеспечает его целесообразное поведение в стационарной случайной среде.

В настоящей работе предложена более точная модель сети связи в качестве нестационарной случайной среды. Особенностью этой модели является то, что она учитывает не только медленные изменения вероятностей получения отказа связаны например с продолжительной перегрузкой, повреждением пучков ветвей или целых узлов коммутации, а также кратковременные колебания этих вероятностей связанны с мгновенной перегрузкой.

Так как никакой из существующих алгоритмов адаптации не обеспечает целесообразного взаимодействия вероятностного автомата с нестационарной случайной средой, предложен новый тип автомата – размытый автомат и алгоритм его адаптации, который для среды этого типа уменьшает среднее число штрафов в составлении с вероятностным автоматом.

3. МОДЕЛЬ ПРОЦЕССА СОСТАВЛЕНИЯ СОЕДИНЕНИЯ В ВИДЕ НЕСТАЦИОНАРНОЙ СЛУЧАЙНОЙ СРЕДЫ

Рассмотрим процесс составления соединения от узла y_i к узлу y_j . Каждому транзитному узлу (направлению) поставим в соответствие

$p_l^{ij}(к)$ – вероятность отказа при составлении соединения от узла y_l^{ij} к узлу y_j.

Если структура сети связи, алгоритмы распределения информации для всех узлов и потоки информации – стационарны, то $p_l^{ij}(к) = p_l^{ij}$ не зависит от момента времени.

Рассмотрим случайный процесс:

$$X_l^{ij}(к) = \begin{cases} 0, \text{ если в момент к алгоритмом управления будет} \\ \quad \text{составлено соединение от } y_i \text{ к } y_j \\ 1, \text{ в противном случае} \end{cases}$$

Очевидно, что

$$P\{X_l^{ij}(к) = 1\} = p_l^{ij}(к)$$

Предположим, что в момент $к_0$ вызов от y_i был проключен к узлу y_l^{ij} и получил отказ $(X_l^{ij}(к_0) = 1)$. Очевидно, что для следующих моментов $к_0 + 1$, $к_0 + 2$, ... вероятность получения отказа при составлении соединения через y_l^{ij} более $p_l^{ij}(к_0+1)$, $p_l^{ij}(к_0+2)$, ...

Определим условную вероятность отказа при составлении соединения от y_l^{ij} к y_j $\tilde{P}_i^{ij}(k_0,к) = P\{X_i^{ij}(k_0+к) = 1 | X_i^{ij}(k_0)$

На рис. 3 изображена типичная зависимость $\tilde{P}_l^{ij}(k_0,к)$ от $к$.

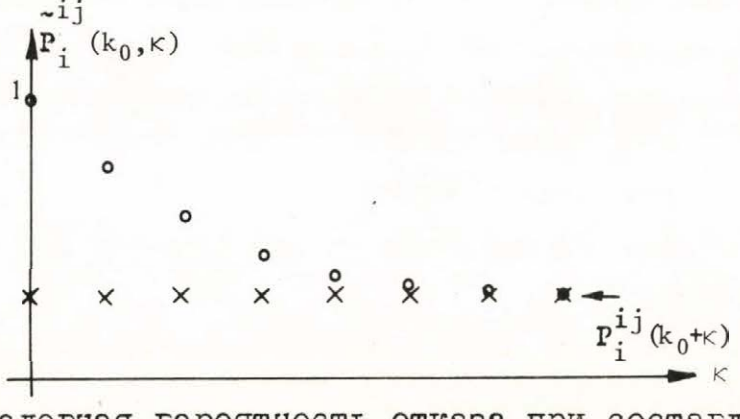


Рис. 3. Условная вероятность отказа при составлении соединения

Аналогично можем определить

$$\tilde{\tilde{P}}_l^{ij}(к_0, к) = P\{X_l^{ij}(к_0+к) = 1 | X_l^{ij}(к_0) = 0\}.$$

Следовательно с точки зрения вероятности отказа процесс составления соединения от y_i к y_j через транзитный узел y_l^{ij} может быть рассматриван в качестве нестационарной цепи Маркова,

для которой матрица переходов

$$C_l^{ij}(к,к_0) = \begin{Vmatrix} c_{00}(к,к_0) & c_{01}(к,к_0) \\ c_{10}(к,к_0) & c_{11}(к,к_0) \end{Vmatrix}$$

где: $к_0$ – момент времени, в котором вызов был последний раз направлен к узлу y_l^{ij}

$к$ – рассматриван момент времени $к \geqslant к_0$

$c_{00}(к,к_0) = P\{\chi_l^{ij}(к) = 0 \mid \chi_l^{ij}(к_0) = 0\} = 1 - \tilde{\tilde{p}}_l^{ij}(к_0, к-к_0)$

$c_{01}(к,к_0) = P\{\chi_l^{ij}(к) = 1 \mid \chi_l^{ij}(к_0) = 0\} = \tilde{\tilde{p}}_l^{ij}(к_0, к-к_0)$

$c_{10}(к,к_0) = P\{\chi_l^{ij}(к) = 0 \mid \chi_l^{ij}(к_0) = 1\} = 1 - \tilde{p}_l^{ij}(к_0, к-к_0)$

$c_{11}(к,к_0) = P\{\chi_l^{ij}(к) = 1 \mid \chi_l^{ij}(к_0) = 1\} = \tilde{p}_l^{ij}(к_0, к-к_0)$

вызывающему узлу y_i и узлу назначения y_j поставил следовательно в соответствие случайную среду

$$E^{ij}(к,к_0) = \langle Y^{ij}, X^{ij}, C^{ij}(к,к_0) \rangle$$

где: $Y^{ij} = \{y_1^{ij}, y_2^{ij}, \ldots, y_n^{ij}\}$

$X^{ij} = \{0,1\}$

$C^{ij}(к,к_0) = \{C_1^{ij}(к,к_0), C_2^{ij}(к,к_0), \ldots, C_n^{ij}(к,к_0)\}$

$C_l^{ij}(к,к_0)$ – матрица переходов цепи маркова соответствующей процессу составления соединения от y_i к y_j через y_l^{ij} .

4. РАЗМЫТЫЙ АВТОМАТ И ЕГО ВЗАИМОДЕЙСТВИЕ С НЕСТАЦИОНАРНОЙ СЛУЧАЙНОЙ СРЕДОЙ

Автомат оптимально взаимодействующий со случайной средой $E^{ij}(к, к_0)$ должен выбирать своё состояние (действие) учитывая:

- среднюю вероятность штрафа $p_l^{ij}(к)$ за каждое действие
- информации о результатах последних действий (реакциях среды), которые можно использовать для оценки текущей вероятности штрафа $\tilde{p}_l^{ij}(к,к)$ или $\tilde{\tilde{p}}_l^{ij}(к,к)$.

Никакой из алгоритмов модификации вероятностного автомата не отвечает этим условиям. Типовые алгоритмы адаптации автомата могут исследовать только медленные изменения случайной среды. Изменяя величину некоторых коэффициентов можно немножко ускорить адаптацию, но в таком случае выбор действия основан на результатах нескольких последних действий и зависит только от кратковременных колебаний вероятностей штрафа. Ниже приведена идея другого типа автомата с переменной структурой и такого алгоритма его модификации, который обеспечает целесообразное взаимодействие со средой $E^{ij}(к,к_0)$.

Элементарный размытый автомат $M(к)$ определяется как

$$M(k) = <A,X,Y,\Gamma_{\tilde{A}}(k)>>$$

где: $A = \{a_1, a_2, \ldots, a_n\}$ - множество состояний автомата

$X = \{0,1\}$ - входной алфавит

$Y = \{y_1, y_2, \ldots, y_n\}$ - выходной алфавит (множество действий автомата)

$\Gamma_{\tilde{A}}(k)$ - размытое состояние автомата - размытое множество в A [4] определяемое функцией принадлежности

$$\mu(k) : A \to [0,1]$$

В каждый момент времени выбор состояния размытого автомата определяется максимальным числом $\mu_i(k)$ ($\mu_i(k) = \mu(a_i,k)$).

$$\forall_{j \neq i} \quad \mu_i(k) > \mu_j(k) \Rightarrow a(k) = a_i$$

Состояние автомата определяет его выход, т.е. для каждого к

$$y(к) = y_i \Leftrightarrow a(к) = a_i .$$

Элементарный размытый автомат может взаимодействовать со случайной средой $E(к,к_0)$. В каждый момент времени происходит тогда модификация функции принадлежности $\mu(к)$ в зависимости от реакции среды.

Представим функцию принадлежности $\mu(к)$ в виде

$$\mu_i(к) = \mu'_i(к)\ (1-\mu''_i(к)),\quad i = 1,2,\ldots,n$$

где: $\mu'(к) : A \to [0,1]$

$\mu''(к) : A \to [0,1]$

Алгоритм модификации в каждый момент времени независимо изменяет величину $\mu'(к)$ и $\mu''(к)$.

$$\mu'(к+1) = T'(\mu'(к), y(к), x(к))$$
$$\mu''(к+1) = T''(\mu''(к), y(к), x(к))$$

где: T' – алгоритм адаптации автомата к медленным изменениям (стационарным характеристикам) среды

T'' – алгоритм адаптации автомата к кратковременным колебаниям характеристик среды вызванным выступлением штрафа или поощрения для какого-нибудь действия выполненного автоматом.

Функции T' и T'' выполняют следующие условия:

если $y(к) = y_l$, то

$$\mu'_m(к+1) = T'(\mu'(к), y_i, x(к)) = \begin{cases} \varphi_1(\mu'_m(к), x(к)), & m = l \\ \mu'_m(к), & m \neq l \end{cases}$$

где: $\varphi_1(\cdot, 0) \in C[0,1]$,

$\varphi_1(1,0) = 1$,

для каждого $\mu \in [0,1)$ $\quad \mu < \varphi_1(\mu,0) < 1$

$\varphi_1(\cdot, 1) \in C[0,1]$,

$\varphi_1(0,1) = 0$,

для каждого $\mu \in (0,1]$ $\quad 0 < \varphi_1(\mu,1) < \mu$

$$\mu''_m(к+1) = T''(\mu''(к), y_i, x(к)) = \begin{cases} \alpha\, x(к), \alpha \in (0,1), & m = l \\ \varphi_2(\mu''_m(к)), & m \neq l \end{cases}$$

где: $\varphi_2 \in C[0,1]$

$\varphi_2(0) = 0$

для каждого $\mu \in (0,1)$ $\quad 0 < \varphi_2(\mu) < \mu$

Выбор функции φ_1 и φ_2 зависит от параметров среды $C(к,к_0)$. Предполагая, что средняя вероятность штрафа за каждое действие не зависит от момента времени примем

$$\varphi_1(\mu'_l(к), x(к)) = \frac{1}{k}[(к-1)\,\mu'_l(к) + 1 - x(к)]$$

Если средняя вероятность штрафа является медленноизменяющейся функцией времени примем, например:

$$\varphi_i(\mu_i'(к), x(к)) = \begin{cases} \frac{1}{k}[(к-1)\,\mu_i'(к) + 1 - x(к)], & к \leq к_m, \\ \frac{1}{k_m}[(к_m-1)\,\mu_i'(к) + 1 - x(к)], & к > к_m. \end{cases}$$

Предполагая, что условная вероятность штрафа $\tilde{p}_i(к_0, к)$ выполняет для каждого к условие

$\beta(1-\varepsilon)[\tilde{p}_i(к_0,0)-p_i(к_0)] \leq \tilde{p}_i(к_0,1)-p(к_0+1) \leq$
$\beta(1+\varepsilon)[\tilde{p}_i(к_0,0)-p_i(к_0)]$

где $\beta \in (0,1)$, $\varepsilon << \beta$ функцию φ_2 можно определить как

$$\varphi_2(\mu_m''(к)) = \beta\,\mu_m''(к)$$

ЗАКЛЮЧЕНИЕ

Размытый автомат с предложенным алгоритмом модификации взаимодействующий со средой $Е(к,к_0)$ выбирает своё действие в зависимости от

- оценки средней вероятности штрафа за каждое действие
- доступной информации об результатах нескольких последних действий, которые используются для оценки текущей величины вероятности штрафа за каждое действие.

Потому можно предполагать, что размытый автомат будет взаимодействовать со средой $Е(к,к_0)$ лучше вероятностного автомата и следовательно размытый автомат использован в системе управления распределением потоков информации в сети связи будет уменьшать среднее число отказов в составлении с вероятностным автоматом.

Л И Т Е Р А Т У Р А

1. В.Г. Лазарев, Г.Г. Саввин – "Сети связи, управление и коммутация", "Связь", 1973
2. Б.Г. Лазарев, Н.Я. Паршенков – "Игровой метод динамического управления сетью связи" в сборнике "Построение управляющих устройств и систем", "Наука", 1974
3. Н.Я. Паршенков – "Использование автоматов с переменной структурой в задаче управления потоками", ИФАК Симпозиум – "Дискретные системы", 1977
4. **L.A. Zadeh – "Fuzzy algorithms", Information and Control, 1965/8/**
5. **K.S. Narendra, M.A.L. Thathachar – "Learning Automata – A Survey", IEEE Trans. on Systems, Man and Cybernetics", July 1974.**

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1978

MICHAL SERVÍT, ZDENĚK FRIŠ, JAN SCHMIDT

AN AUTOMATIC ROUTEING PROGRAM SYSDEB77 FOR PRINTED CIRCUIT BOARDS

Czech Technical University, Dept.of Computers

INTRODUCTION

Modern electronic equipment is based upon the printed circuit board. Components are mounted on a board and interconnections between component pins are made using copper patterns etched on the surfaces of the board. There are two main problems connected with the design of printed circuit boards: where to place the components (placement) and how to form the interconnections (routeing). Computers have been used for many years to assist the solution of both problems [4].

The purpose of this paper is to outline the main features of the SYSDEB77 system [1], [2], [3]. SYSDEB77 is completely capable of automatic routeing on single-sided and double-sided printed circuit boards. The input data files for the SYSDEB77 contain a formalized description of component location and a formalized description of interconnections required. SYSDEB77 automatically produces NC plotter and drill tapes. SYSDEB77 is established on a medium-sized computer, the TESLA 200.

1. STRUCTURE OF SYSDEB77

The SYSDEB77 software consists of three programs (DATA, DESIGN and REALIZATION) using a common data base DAT (Fig.1).

The program DATA checks input files TCB (Type Construction of a Board) and INT (INTerconnection). This program takes over necessary information from a LIB (LIBrary) file.

The program DESIGN performs the ordering of interconnection nets, automatic routeig, optimisation and editing. The editing is controlled by an EDT (EDiT) file.

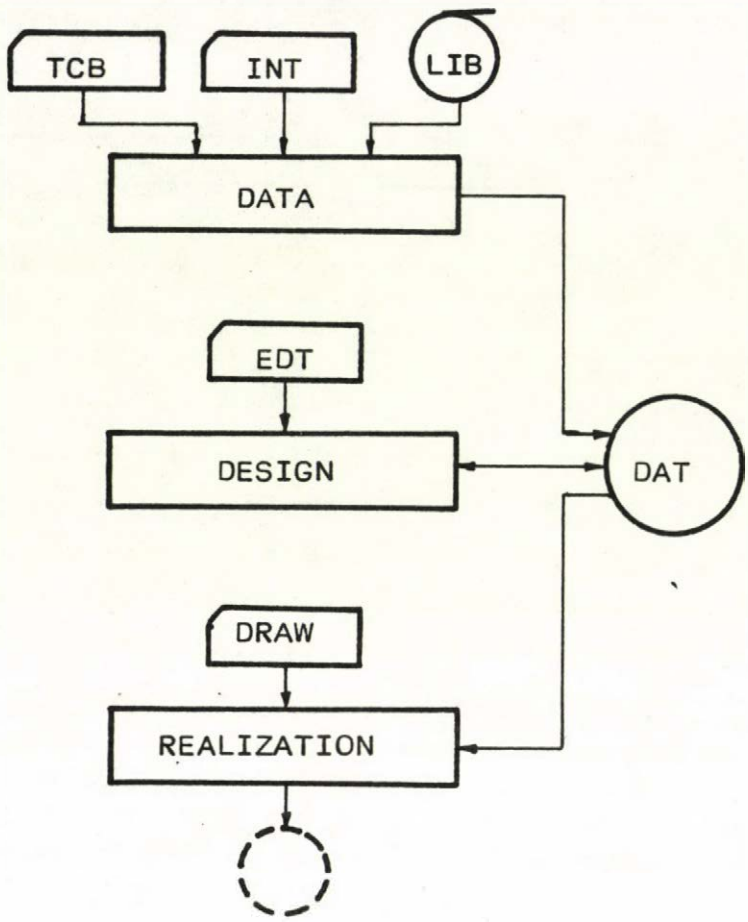


Fig.1

The program REALIZATION produces tapes for NC plotters and NC drilling machines. This program performs the necessary sorting, transformation and editing of graphical information according to the user's requirements specified by a DRAW file.

The SYSDEB77 hardware consists of a TESLA 200 central processor with a 64 k byte (or 128 k byte) core memory, a line printer, paper card reader, paper tape puncher, console typewriter and four magnetic tape units. The TESLA 200 processor has approximately 20 000 operations/s.

2. DATA FILES

SYSDEB77 uses five data files TCB, LIB, INT, EDT, DRAW and data base DAT.

2.1 TCB file

The "Type Construction of a Board" file contains the infor-

mation describing a board: name, size, routeing area, location of components etc. Each component is indicated by its name and defined by its type.

2.2 LIB file

A topological description of component types is contained in the "LIBrary" file. Each component can be divided into several subcomponents indicated by their names. Each subcomponent consists of one or more elements. SYSDEB77 allows the use of 19 different elements: 4 types of tracks, 4 types of pads, 4 types of holes, 4 types of feedthroughs and 3 types of "obstacles".

For example a type RESISTOR consists normally of two subcomponents corresponding to two resistor pins. Each subcomponent consists normally of one feedthrough.

2.3 INT file

The "INTerconnection" file describes all interconnection nets which must be routed on a board. Each net is indicated by its name and defined by a set of subcomponent or component names.

2.4 EDT file

The "EDiT" file allows a designer to remove a track, specify a new track and define a new interconnection net which can be automatically routed.

2.5 DRAW file

The "DRAW" file contains the graphical interpretation of elements, graphical information which has no relation to the routeing area (texts, producers signs etc.), definition of required transformation, specification of an output device etc.

3. DESIGN DESCRIPTION

The design of a printed circuit board by SYSDEB77 is normally performed in three steps.

3.1 Check of TCB and INT files

A designer has to work out files TCB and INT describing his routeing problem before using the System. The program DATA checks both these files from the syntactical and semantical viewpoint. This program supplies the designer with all the necessary information (including the drawing of a board made on a line printer) allowing him to revise files TCB and INT. The revision is usually completed after two runs of program DATA. The result product of this program is stored in the data base DAT.

3.2 Routeing

The run of program DESIGN consists of several steps including: the ordering of nets, routeig, optimisation, the print of drawing on a line printer and storing a result in the data base DAT. A designer can influence the result obtained by chosing several parameters as: sorting - no sorting, optimisation - no optimisation, strategy A - B - F, the increments of weighting function etc. Normally, 100% completion is obtained after the first run of program DESIGN for a medium density boards. In case some connections are missed an editing can be used.

3.3 Production of tapes

A designer has to choice an output device, define the graphical interpretation of elements etc. - that is to form the DRAW file. The program REALIZATION checks this file and produces tapes for the chosen device.

4. ALGORITHMS

4.1 Ordering

Interconnection nets are routed in the order of their priority numbers. The priority number can be specified by a designer or computed by the program DESIGN. The following formula is used for the computation of priority numbers [1]:

$$\text{priority number} = \frac{\text{approximated length of path}}{\text{number of interconnected subcomponents} - 1}$$

4.2 Strategy

The routeing of complete nets is used [4, p.324] . Three strategies (denoted A, B, F) for labelling "start" and "target" objects are available.

4.3 Routeing

A modification of Hoel's "C Algorithm" [5] is used for finding a minimum cost path between a "start" object and a "target" object. Each grid intersection (i.e. two cells in the case of a two-sided board) is stored in one byte. Frontier cells are stored in stacks. An incremental weight function with optional integer nonnegative increments is used for the computing of a path cost. A special retrace procedure is used for removing the unnecessary kinks [1].

4.4 Optimisation

An efficient procedure [3] is used for minimising the number of feedthroughs by means of a topological transformation.

5. RESULTS

SYSDEB77 has been used in production for two years. Most of the boards designed are characterized by the features stated in Table I.

number of layers	2
grid	1.25 x 1.25 mm
pad size	1.5 mm round
track width	0.3 mm
conductor-conductor spacing	0.35 mm minimum
component density	500-800 mm^2 / IC equivalent

Table I

The medium sized printed circuit board from Table II and Fig.2, 3 can serve as an example of routine design. The results are given in Table III.

card type	double-sided printed circuit board
size	236 x 185 mm
routeing area	202.5 x 157.5 mm
number of components	49 IC's, 33 discretes, 4 connectors (55.6 IC equivalents)
component density	573.63 mm^2 / IC equivalent

Table II

task	man hours	CPU time	note
forming of files TCB and INT	10 hours	-	
revision of TCB and INT	3 hours	15 minutes	2 runs of program DATA
automated routeing	-	100 minutes	run of program DESIGN
editing	-	-	no editing was necessary
forming of file DRAW	0.5 hour	-	
revision of file DRAW	-	-	no revision was necessary
production of tapes	-	30 minutes	run of program REALIZATION
total	13.5hours	145 minutes	

Table III

CONCLUSION

We do not claim that SYSDEB77 solves all the problems connected with the automatic routeing on printed circuit boards. However, the results obtained allow us to conclude that SYSDEB77 is a suitable tool for the design of single-sided and double-sided boards.

REFERENCES

[1] Servít,M.-Friš,Z.: Printed Circuit Board Design Automation (in Czech), ČVTS FEL, Prague, 1978.

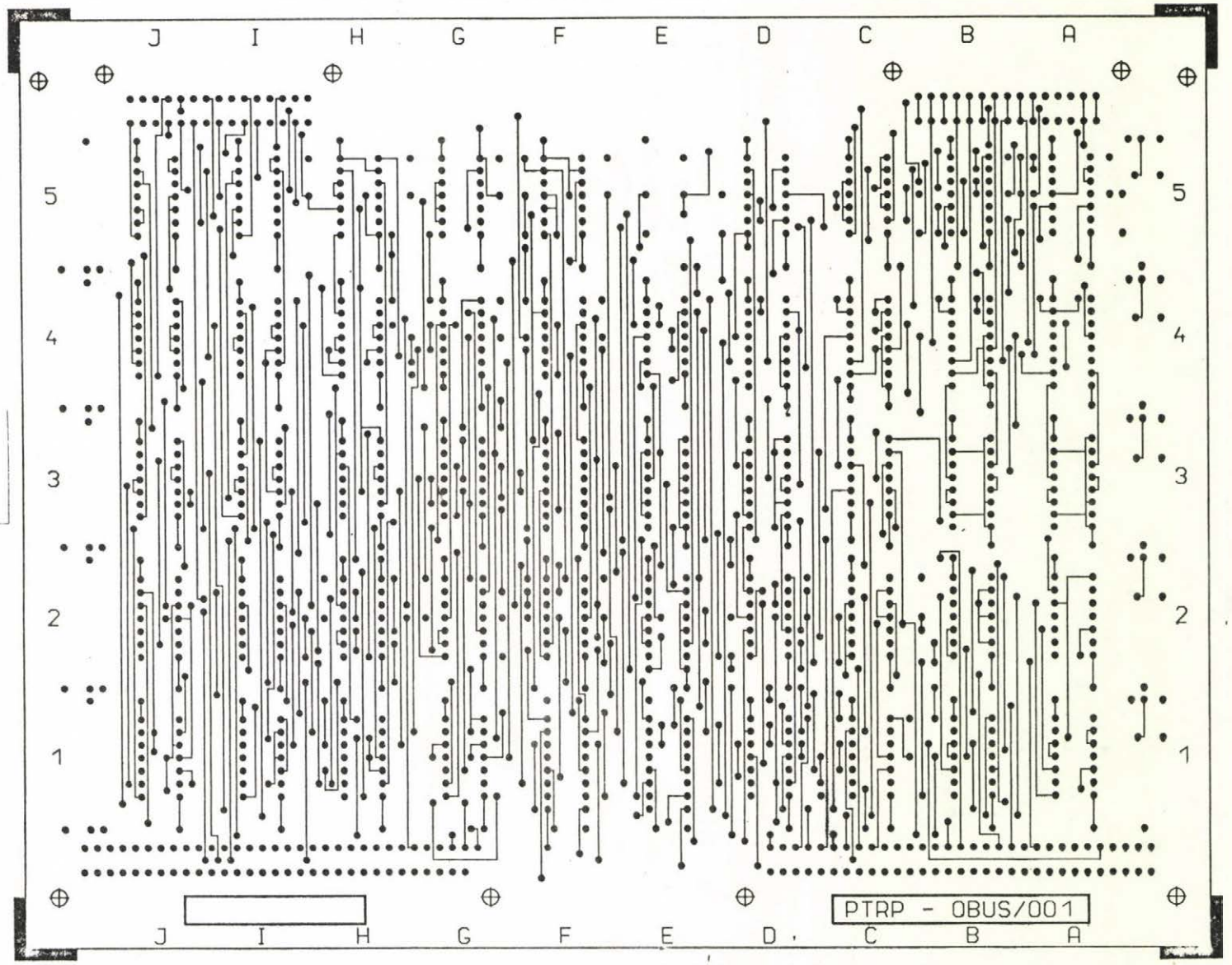
[2] Friš,Z.-Kříž,J.-Servít,M.-Schmidt,J.: Design of Printed Circuit Boards - System SYSDEB77 (in Czech), Automatizace, XXI, pp.9-15, 1978.

[3] Servít,M.: Minimizing the Number of Feedthroughs in Two-Layer Printed Boards, Digital Processes, vol.3, pp.177-183, 1977.

[4] Akers,S.B.: Routing, in "Design Automation of Digital Systems" (ed.Breuer,M.A.), Prentice-Hall, Englewood Cliffs, pp.283-333, 1972.

[5] Hoel,J.H.: Some Variations of Lee's Algorithm, IEEE Trans. on Computers, vol.C-25, pp.19-24, 1976.

STRANA SPOJU - VRSTVA 1



NAVRZENO POCITACEM - SYSDEB77/TESLA 200

Fig.2

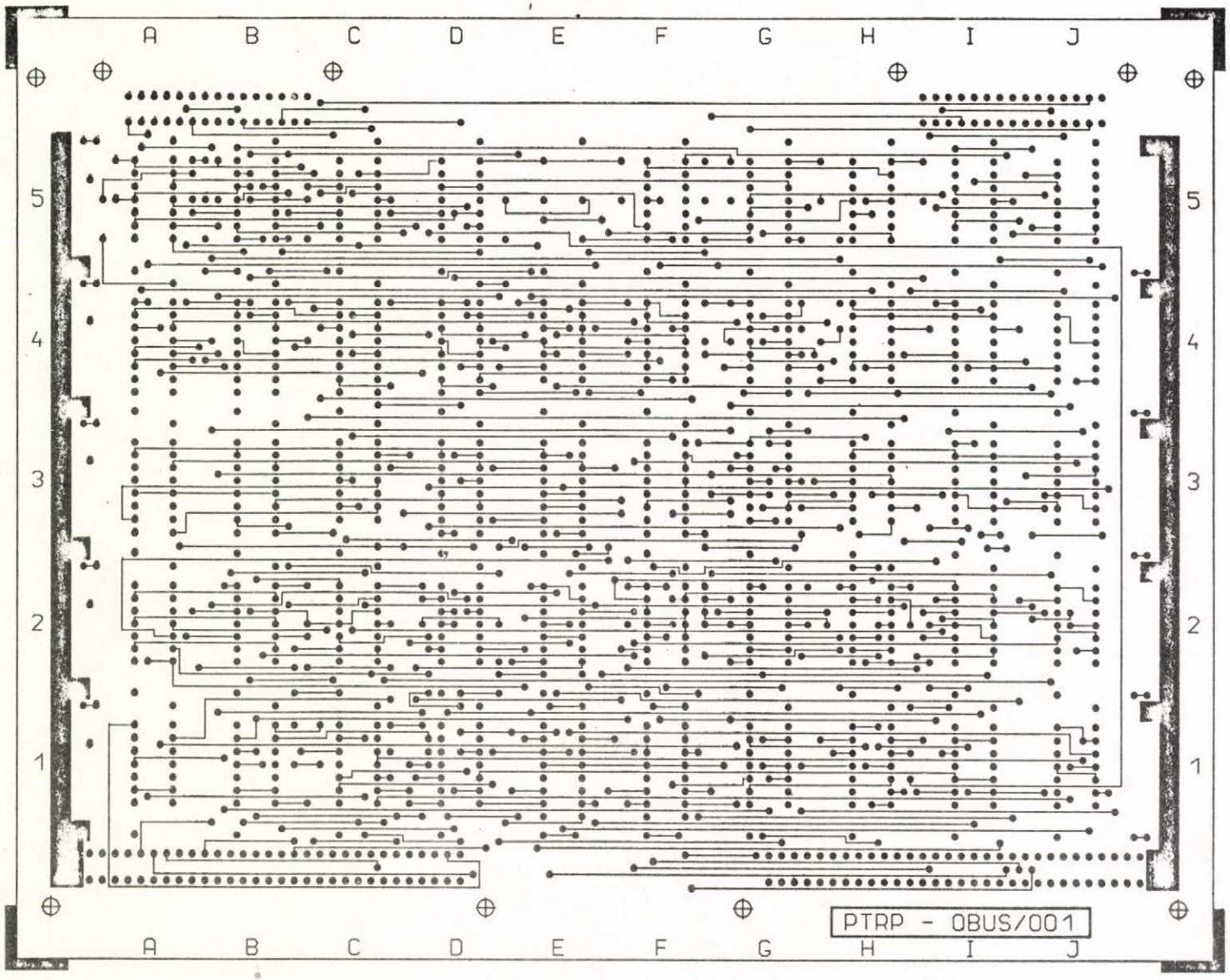


Fig.3

Глава 3: НЕПРЕДСТАВЛЕННЫЕ ДОКЛАДЫ НА НКС 1978 ГОДА

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1978

Г.П.АГИБАЛОВ, Н.В.ЕВТУШЕНКО

ХАРАКТЕРИЗАЦИЯ, УСЛОВИЯ СУЩЕСТВОВАНИЯ И ДРУГИЕ ЗАДАЧИ КАСКАДНОЙ ДЕКОМПОЗИЦИИ КОНЕЧНЫХ АВТОМАТОВ

Сибирский физико-технический институт, Томск, СССР

ВВЕДЕНИЕ

В последние годы вопросы декомпозиции автоматов стали предметом многих исследований [1 - 6] . Объясняется это не только чисто научными интересами - желанием описать структуру любого автомата, но также и соображениями прикладного характера - наряду с каноническим [7] развить декомпозиционный метод синтеза. Далее мы будем иметь дело лишь с каскадными (без обратных связей) декомпозициями конечных автоматов. Такие декомпозиции заслуживают внимания по многим причинам, в частности, в связи с задачами диагностирования дискретных устройств, которые, как известно [8] , решаются значительно проще, именно для структур без обратных связей.

Первые результаты по каскадной декомпозиции конечных автоматов принадлежат Хартманису и Стирнзу, установившим тесную связь между реализациями автоматов каскадными соединениями и системами множеств со свойством подстановки [3] . Они показали, в частности, что если автомат A обладает системой множеств M_1 или парой ортогональных систем M_1 и M_2 такого рода, то он допускает каскадную или, соответственно, её частный вид - параллельную декомпозицию на два автомата A_1 и A_2 такую, что число состояний в A_1 равно числу множеств в системе M_1 , а число состояний в A_2 есть, соответственно, максимальное число состояний в множестве системы M_1 или число множеств системы M_2 . Рассмотрение частного случая систем множеств - разбиений со свойством подстановки позволяет строить точные (т.е. содержащие изоморфные автомату A подавтоматы) каскадные и параллельные декомпозиции любого заданного автомата A .

Системы множеств со свойством подстановки не характеризуют

каскадные декомпозиции автоматов однозначно: при помощи одной и той же системы множеств любого автомата возможно построение, вообще говоря, семейств различных каскадных декомпозиций этого автомата. Методы, которыми пользуются Хартманис и Стирнз, доставляют в каждом случае лишь одну такую декомпозицию. Ниже мы введём понятие сохраняемого нумерованного покрытия и покажем, что в терминах этого понятия могут быть охарактеризованы всевозможные как произвольные, так и точные каскадные, в том числе параллельные и последовательные (ещё один частный вид) декомпозиции любого конечного автомата. Тем самым будет дан способ, которым для любого заданного автомата можно исчерпать все его декомпозиции любого указанного вида. Результаты Хартманиса и Стирнза входят в него как частный случай.

На базе систем множеств со свойством подстановки в работе [4] сформулированы условия разложимости произвольного автомата в каскадное соединение компонент с меньшим числом состояний. Ниже с использованием аппарата сохраняемых нумерованных покрытий устанавливаются необходимые и достаточные условия существования для произвольного автомата декомпозиций перечисленных выше видов. Соответствующие условия, содержащиеся в работах Хартманиса и Стирнза, выводятся из них как следствия.

В статье показывается также, что всякий автомат, за исключением тривиального - тождественного и константного, с двумя состояниями, приводим в том смысле, что всегда существует реализующее его каскадное соединение, в котором ни одна из компонент не реализует данный автомат. Наконец, устанавливается идентичность понятий реализации и моделирования автоматов и тем самым показывается распространимость результатов, относящихся к декомпозициям, определённым в терминах реализации (моделирования), в том числе и результатов данной статьи, на декомпозиции, определённые в терминах моделирования (реализации).

I. ОПРЕДЕЛЕНИЯ И ВСПОМОГАТЕЛЬНЫЕ ПРЕДЛОЖЕНИЯ

Под автоматом, как обычно, понимается набор из пяти объектов

$\langle X, Q, Y, \Psi, \varphi \rangle$, где X, Q и Y суть конечные множества, называемые, соответственно, входным алфавитом, множеством состояний и выходным алфавитом автомата, Ψ и φ - отображения декартова произведения $X \times Q$ в множества Q и Y соответственно. Отображение Ψ называется функцией переходов, отображение φ - функцией выходов.

Функции Ψ и φ, определенные в автомате $\langle X, Q, Y, \Psi, \varphi \rangle$ на парах $(x, q) \in X \times Q$, распространим на пары $(\alpha, q) \in X^* \times Q$, где X^* есть множество всех слов в алфавите X, т.е. всевозможных цепочек конечной длины, составленных из символов алфавита X. Сделаем это индукцией по длине слова α при помощи следующего определения: если $\beta \in X^*$, $x \in X$, $\alpha = \beta x$ и $\Psi(\beta, q)$ уже определено, то $\Psi(\alpha, q) = \Psi(x, \Psi(\beta, q))$ и $\varphi(\alpha, q) = \varphi(x, \Psi(\beta, q))$. Условимся также в следующем обозначении: $\Psi(x, S) = \{\Psi(x, q) : q \in S\}$ для любых $x \in X$ и $S \in \tilde{Q}$. Здесь и далее для произвольного множества M символом $\tilde{M}$ обозначается множество всех конечных подмножеств в M.

Определение [3]. Автомат $A' = \langle X', Q', Y', \Psi', \varphi' \rangle$ реализует автомат $A = \langle X, Q, Y, \Psi, \varphi \rangle$, или A реализуется автоматом A', если существуют отображения $\rho_1 : X \to X'$, $\rho_2 : Q \to \tilde{Q}'$ и $\rho_3 : Y' \to Y$ такие, что
$\forall q \in Q \, \forall x \in X \left[\Psi'(\rho_1 x, \rho_2 q) \subseteq \rho_2 \Psi(x, q) \wedge \forall q' \in \rho_2 q \, (\varphi(x, q) = \rho_3 \varphi'(\rho_1 x, q')) \right]$.
В частности, если ρ_2 есть взаимно однозначное отображение в множество одноэлементных подмножеств из Q', то говорят о точной реализации A автоматом A'.

Определение [3]. Автомат $A' = \langle X', Q', Y', \Psi', \varphi' \rangle$ называется подавтоматом автомата $A = \langle X, Q, Y, \Psi, \varphi \rangle$, если $X' \subseteq X$, $Q' \subseteq Q$ и функции Ψ' и φ' совпадают с Ψ и φ соответственно на области $X' \times Q'$.

Определение. Состояние q_1 автомата $A_1 = \langle X, Q_1, Y_1, \Psi_1, \varphi_1 \rangle$ и состояние q_2 автомата $A_2 = \langle X, Q_2, Y_2, \Psi_2, \varphi_2 \rangle$ называются неотличимыми, если $\forall \alpha \in X^* (\varphi_1(\alpha, q_1) = \varphi_2(\alpha, q_2))$; в противном случае состояния q_1 и q_2 отличимы. Автоматы A_1 и A_2 называются

неотличимыми, если каждое состояние одного из них неотличимо от некоторого состояния другого и наоборот; в противном случае автоматы A_1 и A_2 отличимы. Автомат, в котором все состояния попарно отличимы, называется приведённым . Приведённый автомат, неотличимый от автомата A, называется приведённой формой автомата A. Входной символ x_1 автомата $A_1 = \langle X_1, Q, Y_1, \Psi_1, \varphi_1 \rangle$ и входной символ x_2 автомата $A_2 = \langle X_2, Q, Y_2, \Psi_2, \varphi_2 \rangle$ называются неотличимыми, если $\forall q \in Q(\Psi_1(x_1,q) = \Psi_2(x_2,q) \wedge \varphi_1(x_1,q) = \varphi_2(x_2,q))$; в противном случае символы x_1 и x_2 отличимы. Подавтомат $\langle X', Q', Y', \Psi', \varphi' \rangle$ приведённой формы $\langle X, Q, Y, \Psi, \varphi \rangle$ автомата A, в котором (подавтомате) $Q' = Q$, любые два входных символа отличимы и для каждого $x \in X$ существует неотличимый символ $x' \in X'$, называется сильно приведённой формой автомата A.

Определение [3]. Автомат $A = \langle X, Q, Y, \Psi, \varphi \rangle$ называется гомоморфным образом автомата $A' = \langle X', Q', Y', \Psi', \varphi' \rangle$, если существуют отображения $h_1 : X' \to X$, $h_2 : Q' \to Q$ и $h_3 : Y' \to Y$ такие, что h_1 и h_2 сюръективны (являются отображениями на) и для любых $x' \in X'$ и $q' \in Q'$

$$h_2 \Psi'(x';q') = \Psi(h_1 x'; h_2 q') \quad \text{и} \quad h_3 \varphi'(x';q') = \varphi(h_1 x'; h_2 q').$$

При этом, если отображения h_1, h_2 и h_3 взаимно однозначны, то автоматы A и A' называются изоморфными.

Теорема I. A' реализует A, если и только если сильно приведённая форма автомата A есть гомоморфный образ некоторого подавтомата в A'.

Доказательство утверждения нетрудно получить, опираясь на теорему I.6 [3], если заметить, что, во-первых, в любой реализации приведённого автомата с попарно отличимыми входными символами отображение ρ_1 является взаимно однозначным и, во-вторых, автомат A' реализует A тогда и только тогда, когда A' реализует сильно приведённую форму автомата A. Аналогично доказывается

Теорема 2. A' реализует A точно, если и только если сильно

приведённая форма автомата A изоморфна некоторому подавтомату в A'.

Определение. Автомат вида $\langle X, Q, Q, \Psi, \Psi \rangle$, где $\Psi(x, q) = q$ для всех $(x, q) \in X \times Q$, называется автоматом состояний и обозначается тройкой $\langle X, Q, \Psi \rangle$. Автомат $\langle X, Q, Y, \Psi, \Psi \rangle$, в котором функция выходов не зависит существенно от состояния, т.е. $\Psi(x, q) = \Psi'(x)$ для некоторой функции $\Psi' : X \to Y$, называется автоматом без памяти и обозначается тройкой $\langle X, Y, \Psi' \rangle$.

Определение. Каскадным соединением автоматов $A_1 = \langle X_1, Q_1, Y_1, \Psi_1, \Psi_1 \rangle$ и $A_2 = \langle X_2, Q_2, Y_2, \Psi_2, \Psi_2 \rangle$, где $X_2 = X_1 \times Y_1$ называется автомат $A = \langle X, Q, Y, \Psi, \Psi \rangle$, в котором $X = X_1$, $Q = Q_1 \times Q_2$, $Y = Y_1 \times Y_2$ и для любых $x \in X$, $q_1 \in Q_1$ и $q_2 \in Q_2$

$$\Psi(x, \langle q_1, q_2 \rangle) = \langle \Psi_1(x, q_1), \Psi_2(\langle x, \Psi_1(x, q_1) \rangle, q_2) \rangle,$$
$$\Psi(x, \langle q_1, q_2 \rangle) = \langle \Psi_1(x, q_1), \Psi_2(\langle x, \Psi_1(x, q_1) \rangle, q_2) \rangle.$$

Нетрудно видеть, что всякий автомат $A = \langle X, Q, Y, \Psi, \Psi \rangle$ реализуется каскадным соединением автомата состояний $A_1 = \langle X, Q, \Psi \rangle$ и автомата без памяти $A_2 = \langle X \times Q, Y, \Psi \rangle$, в связи с чем вопросы каскадной декомпозиции автоматов имеет смысл исследовать отдельно для автоматов состояний и для автоматов без памяти. Декомпозиция автоматов без памяти - это самостоятельная задача, выходящая за рамки данной статьи. Будем рассматривать далее, за исключением раздела 4, лишь автоматы состояний. Каскадное соединение любых двух таких автоматов $A_1 = \langle X, Q_1, \Psi_1 \rangle$ и $A_2 = \langle X \times Q_1, Q_2, \Psi_2 \rangle$ будем обозначать $A_1 k A_2$. По определению, это есть автомат $A = \langle X, Q_1 \times Q_2, \Psi \rangle$, в котором $\Psi(x, \langle q_1, q_2 \rangle) = \langle \Psi_1(x, q_1), \Psi_2(\langle x, q_1 \rangle, q_2) \rangle$ для всех $x \in X$, $q_1 \in Q_1$ и $q_2 \in Q_2$. В частности, если функция $\Psi_2(\langle x, q_1 \rangle, q_2)$ не зависит существенно от q_1, автомат A называется параллельным, а в случае существенной независимости Ψ_2 от x - последовательным соединением автоматов A_1 и A_2. Аналогично определяются каскадное, параллельное и последовательное соединения любого числа $n \geq 2$ автоматов. В дальнейшем параллельное соединение автоматов A_1 и A_2 обозначается $A_1 \| A_2$, а их последовательное соединение - $A_1 \cdot A_2$. Автомат $A_1 k A_2$ называется каскад-

ной декомпозицией автомата A, если $A_1 \kappa A_2$ реализует A; в этом случае A_1 и A_2 называются компонентами декомпозиции. Если при этом $A_1 \kappa A_2$ реализует A точно, то автомат $A_1 \kappa A_2$ называется точной каскадной декомпозицией автомата A. Аналогично определяются параллельная и последовательная, в том числе точные, декомпозиции автомата A.

2. ХАРАКТЕРИЗАЦИЯ КАСКАДНЫХ ДЕКОМПОЗИЦИЙ

Определение. Система из непустых подмножеств множества Q, объединение которых есть Q, называется покрытием множества Q. Подмножества, образующие покрытие, называются его блоками. Покрытие с попарно непересекающимися блоками называется разбиением. Покрытие $\pi = \{m_1, \ldots, m_k\}$ множества Q называется сохраняемым в автомате $A = \langle X, Q, \Psi \rangle$, если $\forall x \in X \forall m_i \in \pi \exists m_j \in \pi$ $(\Psi(x, m_i) \subseteq m_j)$.

Заметим, что система множеств у Хартманиса и Стирнза есть покрытие в нашем смысле, а свойство подстановки идентично свойству сохраняемости. Соответственно этому термины „подстановочное разбиение" и „сохраняемое разбиение" являются синонимами. Термин "сохраняемое покрытие" введён Зейгером [1]

Пусть N есть множество натуральных чисел $N = \{1, 2, \ldots\}$. Для любого множества M всякое отображение $h : M \to \tilde{N}$ со свойством

$$\forall p, q \in M \, (p \neq q \to h(p) \cap h(q) = \emptyset)$$

будем называть нумерацией элементов в M. В случае, когда $|h(m)| = 1$ для каждого $m \in M$, нумерация h называется точной.

Рассмотрим произвольное сохраняемое покрытие $\pi = \{m_1, \ldots, m_k\}$ автомата $A = \langle X, Q, \Psi \rangle$. Пусть f_i для каждого $i = 1, \ldots, k$ есть некоторая нумерация состояний в m_i и $f = \{f_1, \ldots, f_k\}$. Условимся далее f называть нумерацией для π, а пару $\langle \pi, f \rangle$ - сохраняемым нумерованным покрытием (сокращённо: СНУП) автомата A. В случае, когда π - разбиение и все нумерации в f точные, пара $\langle \pi, f \rangle$ называется сохраняемым точно нумерованным разбиением (сокращённо: СНУР) данного автомата. Состояние

$q \in m_i$, для которого $f_i(q) \ni l$, будем обозначать f_i^l . Очевидно, f_i^l определяет q однозначно. Пусть также $L_i = \bigcup_{q \in m_i} f_i(q)$ для $i = 1, \ldots, k$ и $L_{\pi f} = \bigcup_{i=1}^{k} L_i$. Ассоциируем с $\langle \pi, f \rangle$ множество $A(\pi, f)$ всевозможных каскадных соединений $A_1 k A_2$, в которых автоматы A_1 и A_2 имеют вид:

$$A_1 = \langle X, \pi, \Psi_\pi \rangle, \quad A_2 = \langle X \times \pi, L_{\pi f}, \Psi_f \rangle$$

и удовлетворяют следующим соотношениям для любых $x \in X, m_i \in \pi$ и $l \in L_i$:

$\Psi_\pi(x, m_i) = m_j$ для некоторого $m_j \in \pi$ такого, что $m_j \ni \Psi(x, m_i)$ (1)

$\Psi_f(\langle x, m_i \rangle, l) = z$ для некоторого $z \in f_j(\Psi(x, f_i^l))$. (2)

По построению, если $\langle \pi, f \rangle$ есть СНУР, то множество $A(\pi, f)$ состоит из единственного каскадного соединения, ассоциированного с данным СНУР.

Теорема 3. Для любого СНУП $\langle \pi, f \rangle$ автомата A всякий автомат $A_1 k A_2 \in A(\pi, f)$ реализует A . Более того, если $\langle \pi, f \rangle$ есть СНУР, то $A_1 k A_2$ реализует A точно.

В самом деле, пусть $Q' = \pi \times L_{\pi f}$. Рассмотрим отображения $\rho_2 : Q \to \tilde{Q}'$ и $\rho_3 : Q' \to Q$ такие, что $\rho_2 q \ni \langle m_i, l \rangle \leftrightarrow f_i^l = q$ и $\rho_3 \langle m_i, l \rangle = f_i^l$. Непосредственно проверяются следующие соотношения для $x \in X$, $q \in Q$ и $\langle m_i, l \rangle \in \rho_2 q$: $q = \rho_3 \langle m_i, l \rangle$ и $\langle \Psi_\pi(x, m_i), \Psi_f(\langle x, m_i \rangle, l) \rangle \in \rho_2 \Psi(x, q)$, из которых, по определению реализуемости, вытекает первое утверждение теоремы. Если $\langle \pi, f \rangle$ есть СНУР и для некоторого $q \in Q$ имеет место $\rho_2 q \ni \langle m_i, l \rangle$ и $\rho_2 q \ni \langle m_j, t \rangle$, то $f_i^l = f_j^t = q$, в силу чего $q \in m_i$, $q \in m_j$ и, следовательно, по свойству разбиения, $m_i = m_j$. Тем самым $f_i^l = f_i^t = q$, откуда, в силу точности f_i , $l = t$. Таким образом, ρ_2 есть отображение в множество одноэлементных подмножеств из Q' . Взаимная однозначность отображения ρ_2 следует из единственности q , для которого $f_i^l = q$. Теорема доказана.

Теорема 4. Для любой каскадной декомпозиции $A_1 k A_2$ автомата A существуют СНУП $\langle \pi, f \rangle$ этого автомата и автомат $A_1 k A_2 \in A(\pi, f)$ такой, что A_1 и A_2 изоморфны подавтоматам автоматов A_1 и A_2 соответственно.

Доказательство. Пусть в условиях теоремы $A_1 = \langle X_1, Q_1, \Psi_1 \rangle$, $A_2 = \langle X_1 \times Q_1, Q_2, \Psi_2 \rangle$ и $\langle X, Q, \Psi \rangle$ есть сильно приведённая форма автомата A. Тогда, в силу теоремы I, $\langle X, Q, \Psi \rangle$ есть гомоморфный образ некоторого подавтомата $A' = \langle X', Q', \Psi' \rangle$ автомата $A_1 \wr A_2$ и, значит, существуют сюръективные отображения $h_1 : X' \to X$ и $h_2 : Q' \to Q$ такие, что для любых $x' \in X'$ и $q' \in Q'$ имеет место

$$h_2 \Psi'(x', q') = \Psi(h_1 x', h_2 q'). \qquad (3)$$

Каждому $x \in X$ сопоставим некоторый элемент $x_0 \in X'$ такой, что $h_1 x_0 = x$. Пусть далее $X_0 = \{x_0 : x \in X\}$, $S = \{s : \exists t (\langle s, t \rangle \in Q')\}$, $T = \{t : \exists s (\langle s, t \rangle \in Q')\}$ и T_0 есть замыкание множества T, определяемое индуктивно следующим образом: I) $T \subseteq T_0$; 2) если $t \in T_0$, $\langle x_0, s \rangle \in X_0 \times S$ и $t' = \Psi_2(\langle x_0, s \rangle, t)$, то $t' \in T_0$; 3) других элементов в T_0 нет. Очевидно, $S \subseteq Q_1$, $T_0 \subseteq Q_2$ и для всех $x \in X_0$, $s \in S$ и $t \in T_0$ справедливо

$$\Psi_1(x_0, s) \in S \quad \text{и} \quad \Psi_2(\langle x_0, s \rangle, t) \in T_0.$$

Поэтому для ограничений Ψ_1^0 и Ψ_2^0 функций Ψ_1 и Ψ_2 на множествах $X_0 \times S$ и $(X_0 \times S) \times T_0$ соответственно тройки $A_1^0 = \langle X_0, S, \Psi_1^0 \rangle$ и $A_2^0 = \langle X_0 \times S, T_0, \Psi_2^0 \rangle$ являются автоматами; более того A_1^0 есть подавтомат в A_1 и A_2^0 – подавтомат в A_2.

Пусть для определённости $S = \{s_1, \ldots, s_k\}$ и $T_0 = \{t_1, \ldots, t_\rho\}$. Образуем в Q подмножества $m_1, \ldots, m_k$ по правилу

$$q \in m_i \leftrightarrow \exists t \in T_0 (h_2 \langle s_i, t \rangle = q), \quad i = 1, \ldots, k. \qquad (4)$$

Из сюръективности h_2 следует, что $\pi = \{m_1, \ldots, m_k\}$ есть покрытие множества Q. Опираясь на равенство (3), непосредственно убеждаемся, что покрытие π сохраняемо в A, причём, если $\Psi_1^0(x_0, s_i) = s_j$, то $\Psi(x, m_i) \subseteq m_j$. Следовательно, можно построить автомат $Q_1 = \langle X, \pi, \Psi_\pi \rangle$, в котором $\Psi_\pi(x, m_i) = m_j$, если и только если $\Psi_1^0(x_0, s_i) = s_j$. Автомат Q_1, как видно, изоморфен автомату A_1^0 и для него верно (I).

Определим следующим образом нумерацию f_i состояний в блоке m_i для каждого $i = \mathrm{I}, \ldots, k$:

$$l \in f_i(q) \leftrightarrow h_2 \langle s_i, t_l \rangle = q, \quad l = 1, \ldots, \rho \qquad (5)$$

Очевидно, $\Lambda_{\pi f} = \{1, \ldots, \rho\}$. Построим автомат $Q_2 = \langle X \times \pi, \Lambda_{\pi f}, \Psi_f \rangle$, приняв

$\Psi_f(\langle x, m_i\rangle, l) = r$, если $\Psi_2^o(\langle x_o, s_i\rangle, t_l) = t_r$.
По построению, A_2 изоморфен автомату $\mathcal{A}_2^o$. Кроме того, если $\Psi_1^o(x_o, s_i) = s_j$, $\Psi_2(\langle x_o, s_i\rangle, t_l) = t_r$ и $l \in L_i$, то, ввиду (3) и (5), $r \in f_j(\Psi(x, f_i^l))$. Таким образом, верно (2) и $A_1 k A_2 \in \mathcal{A}(\pi, f)$.
Теорема доказана.

Теорема 4′. Для любой точной каскадной декомпозиции $\mathcal{A}_1 k \mathcal{A}_2$ автомата $\mathcal{A}$ существует СНУР $\langle \pi, f\rangle$ этого автомата такое, что в ассоциированном с ним каскадном соединении $A_1 k A_2$ компоненты A_1 и A_2 изоморфны подавтоматам автоматов $\mathcal{A}_1$ и $\mathcal{A}_2$ соответственно.

Данная теорема доказывается аналогично теореме 4. При этом вместо теоремы I используется теорема 2. Попарная непересекаемость блоков в π , определяемых по правилу (4), и точность нумераций в f , определяемых по правилу (5), следуют из взаимной однозначности отображения h_2 .

Определение. Автомат $\mathcal{A}_1 k \mathcal{A}_2$ называется покомпонентным расширением автомата $A_1 k A_2$, если A_1 и A_2 изоморфны подавтоматам автоматов $\mathcal{A}_1$ и $\mathcal{A}_2$ соответственно.

Теоремы 3, 4 и 4′ показывают, что множество всех (точных) каскадных декомпозиций любого заданного автомата $\mathcal{A}$ исчерпывается совокупностью всех покомпонентных расширений каскадных декомпозиций, ассоциированных с сохраняемыми (точно) нумерованными покрытиями (соответственно, разбиениями) автомата $\mathcal{A}$.

Лемма I. Множество $\mathcal{A}(\pi, f)$ содержит автомат вида $A_1 \cdot A_2$ тогда и только тогда, когда

$$\forall m_i \in \pi \, \forall l \in L_i \, \exists r \in L_{\pi f} \, \forall x \in X \, \exists m_j \in \pi (\Psi(x, m_i) \subseteq m_j \wedge f_j^r = \Psi(x, f_i^l)). \quad (6)$$

Лемма 2. Множество $\mathcal{A}(\pi, f)$ содержит автомат вида $A_1 \| A_2$ тогда и только тогда, когда

$$\forall x \in X \, \forall l \in L_{\pi f} \, \exists r \in L_{\pi f} \, \forall m_i \in \pi \, \exists m_j \in \pi (\Psi(x, m_i) \subseteq m_j \wedge l \in L_i \rightarrow f_j^r = \Psi(x, f_i^l)). \quad (7)$$

Ввиду данных лемм и теоремы 3, подобно теоремам 4 и 4′ дока-

зываются следующие теоремы.

Теорема 5. Всякая последовательная (параллельная) декомпозиция автомата $\mathcal{A}$ есть покомпонентное расширение некоторого последовательного (соответственно, параллельного) соединения, ассоциированного с некоторым СНУП этого автомата, обладающим свойством (6) (соответственно, (7)).

Теорема 5′. Всякая точная последовательная (параллельная) декомпозиция автомата $\mathcal{A}$ есть покомпонентное расширение последовательного (соответственно, параллельного) соединения, ассоциированного с некоторым СНУР этого автомата, обладающим свойством (6) (соответственно, (7)).

3. УСЛОВИЯ СУЩЕСТВОВАНИЯ СОХРАНЯЕМЫХ НУМЕРОВАННЫХ ПОКРЫТИЙ И КАСКАДНЫХ ДЕКОМПОЗИЦИЙ

Всякий автомат допускает СНУП, в том числе СНУП со свойством (6) или (7). Кроме того, автомат имеет СНУР тогда и только тогда, когда существует в нём сохраняемое разбиение.

Определение. Покрытия π и λ некоторого множества Q называют ортогональными и пишут $\pi \cdot \lambda = 0$, если $\forall m_i \in \pi \, \forall n_j \in \lambda (|m_i \cap n_j| \leq 1)$.

Определение. Множество $\pi \subseteq \tilde{Q}$ называется плотным в автомате $\mathcal{A} = \langle X, Q, \Psi \rangle$, если

$$\forall q \in Q \, \exists n \in \lambda \, \forall x \in X (\Psi(x, q) \in n). \qquad (8)$$

Теорема 6. Автомат $\mathcal{A} = \langle X, Q, \Psi \rangle$ имеет СНУР $\langle \pi, f \rangle$ со свойством (6) и, следовательно, точную последовательную декомпозицию тогда и только тогда, когда существует пара ортогональных разбиений множества Q , одно из которых сохраняемо, а другое плотное в $\mathcal{A}$.

Теорема 7. Автомат $\mathcal{A} = \langle X, Q, \Psi \rangle$ имеет СНУР $\langle \pi, f \rangle$ со свойством (7) и, следовательно, точную параллельную декомпозицию тогда и только тогда, когда для него существует пара ортогональных

сохраняемых разбиений.

Обе теоремы доказываются аналогично, поэтому докажем только первую из них. Введём сначала одно вспомогательное понятие. Пусть $m_i \subseteq Q$, $\lambda \subseteq \widetilde{Q}$, множества в λ упорядочены некоторым образом, скажем, $\lambda = \{r_1, \ldots, r_t\}$, и для каждого $r_j \in \lambda$ имеет место $|r_j \cap m_i| \leq 1$. Определим отображение $f_i : m_i \to \widetilde{N}$ при помощи следующих соотношений:

$$\forall q \in m_i \forall \ell \in N (f_i(q) \ni \ell \longleftrightarrow q \in r_\ell) \qquad (9)$$

Ввиду $|r_j \cap m_i| \leq 1$ для каждого $r_j \in \lambda$, отображение f_i обладает свойством: $q_1 \neq q_2 \to f_i(q_1) \cap f_i(q_2) = \emptyset$, т.е. является нумерацией элементов в m_i . Назовём эту нумерацию порождённой множеством λ . Нетрудно видеть, что если λ есть разбиение Q , то порождённая им нумерация f_i является точной.

Доказательство теоремы 6. Достаточность. Пусть π и λ суть, соответственно, сохраняемое и плотное разбиения в A , $\pi = \{m_1, \ldots, m_k\}$, $\lambda = \{r_1, \ldots, r_t\}$ и $\pi \cdot \lambda = 0$. Зафиксируем в λ указанный порядок блоков и рассмотрим для каждого $m_i \in \pi$ нумерацию f_i , порождённую λ . Ввиду ортогональности π и λ , каждая такая нумерация существует. Её точность следует из свойства разбиения λ . Пусть $f = \{f_1, \ldots, f_k\}$. Таким образом, $\langle \pi, f \rangle$ есть СНУР автомата A . Покажем, что оно обладает требуемым свойством (6). В самом деле, пусть $m_i \in \pi$, $\ell \in \Sigma_i$ и $x, x' \in X$. Рассмотрим некоторый блок $m_j \in \pi$, для которого $\psi(x, m_i) \subseteq m_j$ и, следовательно, $\psi(x, f_i^\ell) \in m_j$. Пусть $\psi(x, f_i^\ell) = f_j^\tau$ для некоторого $\tau \in \Sigma_j$. Тогда, по свойству сохраняемого разбиения, $\exists m_3 \, \psi(x'; m_i) \subseteq m_3$ и, ввиду (9) и плотности λ , $\psi(x'; f_i^\ell) = f_3^\tau$. Этим в силу произвольности m_i , ℓ , x и x' , доказана справедливость (6). Достаточность установлена.

Необходимость. Образуем λ как совокупность множеств $\{f_j^\tau : m_j \in \pi\}$ для $\tau \in \Sigma_{\pi f}$. Поскольку для каждого $q \in Q$ найдутся $m_j \in \pi$ и $\tau \in \Sigma_j$ со свойством $f_j^\tau = q$, то λ есть покрытие Q . Кроме того, в силу свойства нумерации, никакие два состояния из одного блока разбиения π не попадут в один блок покрытия

λ. Следовательно, π и λ ортогональны. В силу же точности каждой нумерации f_j и благодаря свойству разбиения π, различные блоки в λ не пересекаются, т.е. λ есть разбиение множества Q. Убедимся в плотности λ. Для этого рассмотрим произвольное состояние $q \in Q$. Выберем для него $r \in \mathcal{L}_{\pi f}$ из условия:

$$\forall x \in X \, \exists m_j \in \pi \, (\psi(x,q) \in m_j \wedge f_j^r = \psi(x,q)) . \qquad (10)$$

Ввиду (6), требуемое r обязательно найдется. Построим для него множество $n = \{f_j^r : m_j \in \pi\}$. Очевидно, $n \in \lambda$. Для произвольного $x \in X$ рассмотрим состояние $\psi(x,q)$. Ввиду (10), найдётся блок $m_j \in \pi$ такой, что $\psi(x,q) \in m_j$ и $\psi(x,q) = f_j^r$. Следовательно, $\psi(x,q) \in n$ и, в силу произвольности x, верно (8). Тем самым λ действительно плотное. Необходимость установлена. Теорема 6 доказана.

Аналогично теоремам 6 и 7 доказываются следующие теоремы.

Теорема 8. Если на множестве состояний автомата A существует пара ортогональных покрытий π и λ, одно из которых (π) сохраняемо, а другое (λ) плотное в A, то автомат A имеет СНУП $\langle \pi, f \rangle$ со свойством (6), в котором $|\mathcal{L}_{\pi f}| = |\lambda|$, и, следовательно, допускает последовательную декомпозицию, в которой первый автомат содержит ($|\pi|$) состояний, а второй – ($|\lambda|$) состояний.

Теорема 9. Если автомат A имеет два ортогональных сохраняемых покрытия π и λ, то он имеет СНУП $\langle \pi, f \rangle$ со свойством (7), в котором $|\mathcal{L}_{\pi f}| = |\lambda|$, и, следовательно, допускает параллельную декомпозицию, в которой один автомат содержит $|\pi|$ состояний, а другой – $|\lambda|$ состояний.

Наконец, из теорем 3 и 4′ вытекает, что автомат A допускает точную каскадную декомпозицию, в которой первый автомат имеет n состояний, тогда и только тогда, когда для него существует сохраняемое разбиение мощности n.

4. РЕАЛИЗАЦИЯ И МОДЕЛИРОВАНИЕ

Определение [1,2]. Автомат $A' = \langle X', Q', Y', \Psi', \varphi' \rangle$ моделирует автомат $A = \langle X, Q, Y, \Psi, \varphi \rangle$, или A моделируется автоматом A', если существуют отображения $\mu_1 : X \to X'$, $\mu_2 : Q \to Q'$ и $\mu_3 : Y' \to Y$ такие, что

$$\forall q \in Q \, \forall \alpha \in X^* \left[\varphi(\alpha, q) = \mu_3 \varphi'(\mu_1 \alpha, \mu_2 q) \right].$$

Здесь и далее для любого отображения вида $\mu : U \to V$ и для любого слова $\alpha = u_1 u_2 \dots u_\ell \in U^*$ через $\mu\alpha$ обозначается слово $\mu(u_1)\mu(u_2)\dots\mu(u_\ell) \in V^*$.

Теорема 10. A' моделирует A, если и только если A' реализует A.

Для доказательства достаточности следует принять [3] $\mu_1 = \rho_1$, $\mu_3 = \rho_3$ и для любого $q \in Q$ положить $\mu_2 q = q'$ для некоторого $q' \in \rho_2 q$, а для доказательства необходимости достаточно принять $\rho_1 = \mu_1$, $\rho_3 = \mu_3$ и для любого $q \in Q$ положить $\rho_2 q = S_q$, где $S_q \subseteq Q'$ и $q' \in S_q \leftrightarrow \forall \alpha \in X^* \left[\varphi(\alpha, q) = \mu_3 \varphi'(\mu_1 \alpha, q') \right]$.

5. ПРИВОДИМОСТЬ

В алгебраической теории автоматов одним из центральных понятий является $\mathfrak{z}$-неприводимость.

Определение [1,2]. Автомат A называется $\mathfrak{z}$-неприводимым, если всякий раз, когда он моделируется каскадным соединением двух автоматов A_1 и A_2 с полугруппами S_1 и S_2, он моделируется либо автоматом полугруппы S_1, либо автоматом полугруппы S_2.

Однако для целей структурной теории автоматов более естественным представляется понятие неприводимости.

Определение [1]. Автомат A называется неприводимым, если всякий раз, когда он моделируется каскадным соединением двух автоматов A_1 и A_2, он моделируется либо автоматом A_1, либо автоматом A_2; в противном случае автомат A приводим.

Заметим, что, ввиду теоремы 10, в этих определениях вместо

„моделируется" можно читать „реализуется". На несовпадение понятий $\mathfrak{I}$-неприводимости и неприводимости указано Арбибом [1]. Крон и Роудз [1,2] охарактеризовали класс всех $\mathfrak{I}$-неприводимых автоматов. Его составляют автоматы простых групп и тождественно-возвратные автоматы с двумя состояниями. (Последние определяются ниже). Аналогичная задача возникает в связи с понятием неприводимости.

Условимся для любого входного слова α автомата $\langle X, Q, \Psi \rangle$ через Ψ_α обозначать отображение из Q в Q такое, что $\Psi_\alpha(q) = \Psi(\alpha, q)$ для любого $q \in Q$.

Определение. Автомат $A = \langle X, Q, \Psi \rangle$ называется тождественно возвратным (кратко: ТВ-автоматом), если каждое его отображение Ψ_x, $x \in X$, тождественно или константно. Автомат A называется тождественным (константным), если все его отображения Ψ_x, $x \in X$, тождественны (соответственно, константны и равны). Автомат A называется перестановочно-возвратным (кратко: ПВ-автоматом), если каждое его отображение Ψ_x, $x \in X$, есть подстановка на Q или константно. Наконец, автомат A называется перестановочным (кратко: П-автоматом), если все его отображения Ψ_x, $x \in X$, суть подстановки на множестве Q.

Теорема II. Всякий автомат A с $n \geq 2$ состояниями, не являющийся при $n = 2$ тождественным или константным, приводим. Тождественные и константные автоматы с двумя состояниями неприводимы.

Доказательство. Рассмотрим автомат $\mathcal{A}_n = \langle \hat{X}, \hat{Q}, \hat{\Psi} \rangle$, в котором $\hat{X} = \{x_1, \ldots, x_n\}$, $\hat{Q} = \{1, \ldots, n\}$ и $\hat{\Psi}_{x_i} = (1\ 2 \ldots i)$ - цикл первых i состояний, $i = 1, \ldots, n$. Сформулируем без доказательства, ввиду его громоздкости, следующую лемму.

Лемма 3. Всякий автомат с n состояниями реализуется каскадным соединением автоматов $\mathcal{A}_n$ и ТВ-автоматов с двумя состояниями.

В силу данной леммы, теорема 9 будет доказана, если мы покажем её справедливость для автомата A с n состояниями, реализуемого автоматом $\mathcal{A}_n$ для $n > 2$ или ПВ-автоматом с двумя состояниями. Пусть сначала $n > 2$ и $A = \langle X, Q, \Psi \rangle$ есть произвольный подавтомат в $\mathcal{A}_n$, для которого $Q = \hat{Q}$ и $x_n \in X$. (Нетрудно заметить, что все другие автоматы, реализуемые $\mathcal{A}_n$, либо изоморфны некоторому подавтомату в $\mathcal{A}_n$ указанного вида, либо реализуются автоматом $\mathcal{A}_{n-1}$). Рассмотрим автомат $A' = \langle X, Q', \Psi' \rangle$ в котором $Q' = \{1, \ldots, n+1\}$, функция Ψ' совпадает с Ψ на области $X \times Q$ и $\Psi'(x, n+1) = n+1$ для любого $x \in X$. Очевидно, A' реализует A. Для автомата A' построим СНУП $\langle \pi, f \rangle$ следующим образом: $\pi = \{m_1, m_2\}$, $m_1 = m_2 = Q'$, $f_1^l = f_2^l = l$ для $l < n$, $f_1^n = f_2^{n+1} = n$ и $f_1^{n+1} = f_2^n = n+1$. Рассмотрим каскадное соединение $A_1 k A_2 \in A'(\pi, f)$, где $A_1 = \langle X, \pi, \Psi_\pi \rangle$ и $A_2 = \langle X \times \pi, Q', \Psi_f \rangle$, такое, что $\Psi_{\pi, x} = (m_1, m_2)$ для всех $x \in X$. Ввиду (1) и (2), непосредственно проверяется, что $\Psi_{f, \langle x_n, m_1 \rangle} = (1\ 2 \ldots n-1\ n+1\ n)$ и $\Psi_{f, \langle x_n, m_2 \rangle} = (1\ 2 \ldots n)$. В силу теоремы 3, автомат $A_1 k A_2$ реализует A и вместе с ним A. Нетрудно убедиться также, что A_1 не реализует A, и всякий подавтомат в A_2, имеющий не менее n состояний, является в действительности сильно связным П-автоматом с $n+1$ состояниями и, в силу теоремы 2 [6], не имеет сохраняемого разбиения мощности n. Поэтому, ввиду следующей леммы, вытекающей из определений реализации и сохраняемого разбиения, A_2 не реализует A и, следовательно, A приводим.

Лемма 4. Во всяком автомате, реализующем автомат с n состояниями, существует подавтомат, имеющий сохраняемое разбиение мощности n.

Пусть теперь автомат $A = \langle X, Q, \Psi \rangle$ реализуется ПВ-автоматом с двумя состояниями и не является тождественным или константным Тогда возможны следующие два случая.

1) $A = \langle X, \{1,2\}, \Psi \rangle$ и существует $x \in X$, что $\Psi_x = (2\ 1)$. Рассмотрим СНУП $\langle \pi, f \rangle$ этого автомата, в котором $\pi = \{m_1, m_2, m_3\}$, $m_1 = m_2 = m_3 = \{1,2\}$, $\Delta_{\pi f} = \{1,2,\ldots,6\}$, $f_1^1 = f_2^3 = f_3^5 = 1$ и $f_1^2 = f_2^4 = f_2^3 = 2$. Пусть $A_1 k A_2 \in A(\pi, f)$, $A_1 = \langle X, \pi, \Psi_\pi \rangle$, $A_2 = \langle X \times \pi, \Delta_{\pi f}, \Psi_f \rangle$,

причём $\psi_{\pi,x}=(m_1\ m_2\ m_3)$, $\psi_{f,\langle x,m_1\rangle}=(1\ 4\ 2\ 3\ 5)$, $\psi_{f,\langle x,m_2\rangle}=(2\ 3\ 6\ 4\ 5)$ и $\psi_{f,\langle x,m_3\rangle}=(1\ 3\ 5\ 2\ 6)$. Непосредственно проверяется, что Q_1 и Q_2 не реализуют A. Следовательно, A приводим.

2) $A=\langle X,\{1,2\},\psi\rangle$ и существуют x_1, $x_2\in X$, что $\psi(x_1,q)=1$ и $\psi(x_2,q)=2$ для любого $q=1,2$. Рассмотрим СНУП $\langle\pi,f\rangle$, в котором $\pi=\{m_1,m_2,m_3\}$, $m_1=\{1,2\}$, $m_2=\{2\}$, $m_3=\{1\}$, $\Lambda_{\pi f}=\{1,2,3\}$, $f_1^1=f_3^1=f_3^2=1$ и $f_1^2=f_1^3=f_2^1=f_2^3=2$. Пусть

$Q_1 k Q_2\in A(\pi,f)$, $Q_1=\langle X,\pi,\psi_\pi\rangle$, $Q_2=\langle X\times\pi,\Lambda_{\pi f},\psi_f\rangle$, причём

$$\psi_{\pi,x_1}=\begin{pmatrix}m_1 & m_2 & m_3\\ m_3 & m_3 & m_1\end{pmatrix},\quad \psi_{\pi,x_2}=\begin{pmatrix}m_1 & m_2 & m_3\\ m_2 & m_1 & m_2\end{pmatrix},\quad \psi_{f,\langle x_2,m_2\rangle}=\begin{pmatrix}1 & 2 & 3\\ 2 & 3 & 3\end{pmatrix}$$

$$\psi_{f,\langle x_1,m_1\rangle}=\psi_{f,\langle x_1,m_2\rangle}=\psi_{f,\langle x_2,m_3\rangle}=\begin{pmatrix}1 & 2 & 3\\ 1 & 1 & 2\end{pmatrix},\quad \psi_{f,\langle x_2,m_1\rangle}=\psi_{f,\langle x_2,m_3\rangle}=\begin{pmatrix}1 & 2 & 3\\ 1 & 3 & 1\end{pmatrix}$$

Непосредственно проверяется, что Q_1 и Q_2 не реализуют A и, следовательно, A приводим.

Наконец, в приводимости константных и тождественных автоматов с $n>2$ состояниями и в неприводимости таких автоматов с двумя состояниями легко убедиться самостоятельно. Теорема доказана

6. К ДЕКОМПОЗИЦИИ СИЛЬНО СВЯЗНЫХ П-АВТОМАТОВ

Пусть всюду далее $A=\langle X,Q,\psi\rangle$ есть сильно связный П-автомат с n состояниями и G – его группа.

Теорема 12. Автомат A реализуется каскадным соединением меньших (с меньшим числом состояний) автоматов, если и только если существуют нормальные делители G_0, G_1, ..., G_k группы G такие, что $G=G_0\triangleright G_1\triangleright\cdots\triangleright G_k=1$ и для любого $i=1,\dots,k$ фактор-группа G_{i-1}/G_i изоморфна группе подстановок степени меньше n.

Доказательство. Достаточность следует из теоремы Зейгера [1] и теоремы в [9]. Необходимость доказывается индукцией по длине каскадного соединения с применением на каждом шаге индукции формулируемой ниже леммы 5. Пусть π есть сохраняемое в A покрытие, состоящее из различных блоков некоторой одинаковой мощности ρ. Рассмотрим автомат $A'=\langle X,\pi,\psi'\rangle$, в котором

$\psi'(x,m_i)=m_j \longleftrightarrow \psi(x,m_i)=m_j$ для всех $x\in X$ и $m_i\in\pi$.

Пусть G' есть группа автомата A'.

Лемма 5. G' есть гомоморфный образ группы G и, если $G' \neq G$ то существуют нетривиальное разбиение множества Q на классы $Q_1, \ldots, Q_d$ и нормальные делители $H_0, H_1, \ldots, H_d$ группы G такие, что $d \mid n$, $|Q_i| = \frac{n}{d} \leq p$ для каждого $i = 1, \ldots, d$, $G \triangleright H_0 \triangleright H_1 \triangleright \ldots \triangleright H_d = 1$, каждый класс Q_i есть область транзитивности H_0, каждый делитель H_i для $i > 0$ оставляет на месте символы из $Q_1 \cup \ldots \cup Q_i$ и каждый фактор H_{i-1}/H_i изоморфен группе подстановок множества Q_i.

Из теоремы 12 вытекают следующие следствия. 1. Если A реализуется некоторым каскадным соединением меньших автоматов, то автомат группы G изоморфен одному из таких соединений.
2. Если G есть S_n или A_n для $n \neq 4$, или если n - простое число, то A не реализуется каскадным соединением меньших автоматов.

ЛИТЕРАТУРА

1. Алгебраическая теория автоматов, языков и полугрупп. Под редакцией М.А.Арбиба. Изд-во „Статистика", М.,1975, 336 с.
2. Калман Р., Фалб П., Арбиб М. Очерки по математической теории систем. Изд-во „Мир", М., 1971, 400 с.
3. Hartmanis J., Stearns R.E. The Algebraic Structure Theory of Sequential Machines. Prentice-Hall, N.J., 1966.
4. Dömösi P. On Superpositions of Automata. Acta Cybernetica, Tom.2, Fasc.4, Szeged, 1975, pp.335-343.
5. Агибалов Г.П., Ванина Н.В. О сложности декомпозиции автоматов методом Зейгера. АВТ, № 5, 1974, с. 6-11.
6. Агибалов Г.П., Евтушенко Н.В. К декомпозиции конечных автоматов. АВТ, № 5, 1976, с. 15-21.
7. Глушков В.М. Синтез цифровых автоматов. ФМ, М.,1962, 476 с.
8. Чжен Г., Мэннинг Е., Метц Г. Диагностика отказов цифровых вычислительных систем. Изд-во „Мир", 1972, 232 с.
9. Евтушенко Н.В. О декомпозиции конечных автоматов в каскадное соединение перестановочных автоматов. В кн.: Проблемы проектирования и применения дискретных систем в управлении Минск, 1977, с. 57-59.

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1978

Амбарцумян А.А., Потехин А.И.

ПРОГРАММИРУЕМЫЙ ЛОГИЧЕСКИЙ КОНТРОЛЛЕР НА ОСНОВЕ ПЕРЕСТРАИВАЕМЫХ СТРУКТУР

Институт проблем управления АН СССР

I. ПОСТАНОВКА ЗАДАЧИ

Последние достижения интегральной технологии, в частности в области производства программируемых устройств в элементном базисе дискретной техники таких как ПЗУ, ППЗУ, ПЛМ и микропроцессоры, привели к появлению принципиально новых подходов к построению дискретных устройств для автоматизированных систем управления.

В настоящей работе ставится задача обосновать аппаратурно-программный принцип реализации дискретных устройств и предлагается структура программируемого контроллера на перестраиваемых средах, реализующая этот принцип.

2. Анализ возможностей аппаратурной и программной реализации дискретных управляющих устройств.

Построение дискретных управляющих устройств обычно осуществляется двумя способами: аппаратурным и программным.

Аппаратурная реализация устройства управления представляет собой автоматическое устройство с так называемой жесткой логикой, реализующей единственную систему логических функций в соответствии с заданным алгоритмом управления.

В качестве примера можно привести мощный класс устройств, называемые устройствами цикловой автоматики, для управления такими объектами, как металлорежущие станки, обрабатывающие центры, линии из агрегатных головок, конвеерные линии, транспортные линии, погрузочно-разгрузочные механизмы и т.д.

Аппаратурная реализация систем управления перечисленных объектов осуществляется как правило на основе релейно-контактных схем (РКС).

Программная реализация устройства управления осуществляется на базе управляющих ЭВМ путем программного вычисления логи-

ческих функций, описывающих алгоритм управления объектом. Характерной особенностью управляющих алгоритмов перечисленными объектами является, во-первых, битовый характер входной и выходной информации устройства управления, и, во-вторых, булевый вид вычислений. Рассмотрим основные достоинства и недостатки аппаратурной (в виде РКС) реализации рассматриваемого класса алгоритмов управления и программной реализации на базе УВМ традиционной структуры.

Достоинства и недостатки с точки зрения пользователя (потребителя, эксплуатационщика) и с точки зрения разработчика устройства управления можно разделить соответственно на внешние и внутренние.

В соответствии с этим вначале остановимся кратко на перечислении основных достоинств и недостатков аппаратурной реализации в виде РКС.

К внешним достоинствам РКС можно отнести:

- простота проектирования РКС, заключающаяся в том, что задача описания алгоритма управления сводится к его изображению в виде РКС;
- широкие языковые возможности РКС, заключающиеся в простоте представления параллельности при управлении различными объектами, режимов прерывания, блокировок и т.д.;
- простота внесения изменений в РКС;
- доступность в силу своей наглядности широкому кругу специалистов (инженерам, электрикам, технологам) и как следствие этого простота эксплоатации, ремонта и настройки.

К внутренним достоинствам РКС следует отнести:

- простота проектирования устройства управления по записи алгоритма управления в виде РКС;
- параллельное вычисление всех логических функций, осуществляемое РКС, обеспечивает предельное быстродействие устройства управления.

К недостаткам (внешним) РКС можно отнести:

- трудности реализации арифметических вычислений, необходимость которых может возникнуть при управлении объектом;
- отсутствие универсальности, что приводит к необходимос-

ти использования для каждого объекта управления своей РКС, при этом полностью отсутствует их взаимозаменяемость;

- высокая стоимость изготовления и эксплуатации;
- большие габариты и вес, что связано с особенностью релейно-контактных элементов.

К внутренним недостаткам РКС можно отнести:

- трудности внесения исправлений в устройство управления, необходимость которых возникает при внесении изменений в технологический цикл работы объекта управления;
- трудности реализации РКС на современных полупроводниковых элементах, обусловленных различными физическими свойствами релейно-контактных и полупроводниковых элементов.

Перечисленные недостатки РКС обусловили постепенный отказ от построения устройств управления на базе РКС и переход от аппаратурной реализации алгоритмов управления к программной реализации. В связи с этим представляет,интерес рассмотрения возможностей программной реализации алгоритмов управления на базе мини-УВМ, структура которых в основном копирует структуру ЭВМ общего назначения.

К внешним достоинствам использования в качестве устройства управления УВМ можно отнести:

- универсальность устройства управления;
- ускорение и удешевление проектирования, обусловленное тем, что обычная задача проектирования в случае аппаратурной реализации (получение РКС, создание макета, его отладка и т.д) сводится к задаче программирования с последующей отладкой программы.

Что касается разработки собственно УВМ, то здесь накоплен значительный опыт проектирования архитектуры ЭВМ общего назначения.

К недостаткам использования УВМ можно отнести следующие.

К внешним недостаткам:

- при использовании РКС в качестве языка записи алгоритма управления, обладающего вышеуказанными достоинствами, возникают трудности при программировании, вследствие различных способов вычислений булевых функций, осуществляемых РКС (параллельно) и в УВМ (последовательно);

- необходимость в наличии специалистов-программистов и как следствие этого возникают ряд трудностей при взаимопонимании между ними с одной стороны и инженерами-электриками, технологами, с другой стороны;

- низкое быстродействие (большое время реакции) УВМ, обусловленное принципиально последовательной дисциплиной вычислений;

- низкая надежность УВМ при её использовании в промышленных условиях;

- высокая стоимость.

К внутренним недостаткам УВМ следует отнести:

- пословный принцип обработки информации, присущий традиционным УВМ, приводит к неэффективному использованию оборудования УВМ (памяти, устройств ввода-вывода) при битовом характере данных об объекте;

- необходимость наличия развитого мат.обеспечения, а также операционной системы УВМ для получения эффективной программы и осуществление различного рода прерываний, блокировок в процессе управления объектом;

- необходимости наличия в УВМ оперативного запоминающего устройства (ОЗУ) вследствие последовательного вычисления функций.

Вышеперечисленные недостатки РКС и УВМ по нашему мнению, послужили причиной поиска новых принципов построения устройств управления. В настоящее время в США, ФРГ, Франции (см., например, [1]) появился ряд оригинальных архитектурных решений построения УВМ, в основном специализированных на вычисление булевых функций. Эти логические машины получили название программируемые контроллеры (ПК). Основные отличия ПК от традиционных УВМ продемонстрирует на примере ПК, разработанного американской фирмой MODICON [1] , упрощенная блок-схема которого изображена на рис.I

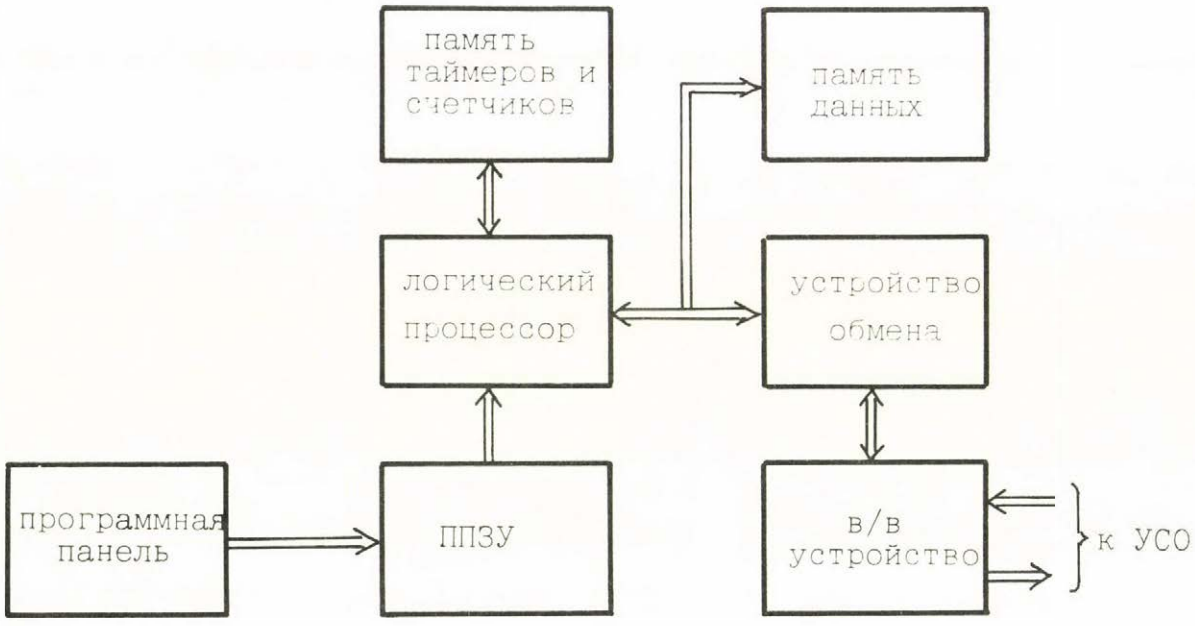


Рис.I.

Основные особенности ПК состоят в следующем:

I) при программировании задачи нет понятия команды. В качестве элемента программы используется строка РКС, содержащая 4 контакта, соединенных произвольным образом;

2) битовое представление данных;

3) разделение памяти ПК на память для программы и память для данных, при этом память программы используется только для считывания информации, что позволяет её реализовать в виде ПЗУ (ППЗУ);

4) специализированный логический процессор для обработки одной строки РКС;

5) сканирование по памяти программы является основным способом функционирования;

6) безадресность строк (отсутствие условного перехода);

7) простота наращивания ПК по мощности (память, программа, вход-выходные регистры);

8) введение простых арифметических подпрограмм, результаты которых используются в основной программе как логические условия.

ПК получили широкое распространение, особенно в областях промышленности с тяжелыми условиями эксплуатации. Тем не менее можно выделить следующие недостатки ПК типа MODICON

I. Низкое быстродействие (большое время реакции) ПК, обусловленное последовательной дисциплиной вычислений булевых функций, при этом время вычисления зависит от длины программы.

2) трудности распараллеливания вычислений, обусловленных взаимозависимостью логических функций, и его реализации в ПК, приводят к тому, что скорость вычисления невозможно увеличить за счёт наращивания оборудования. Эти недостатки ПК ограничивают их применение там, где требуется высокое быстродействие (малое время реакции), например, в случае электронной технологии, управление лазером, генерации тестов для динамической проверки логических схем и т.д.

3. Аппаратурно-программный подход к реализации дискретных устройств управления.

Как показал вышепроведенный анализ и аппаратный и программный подход к реализации дискретных устройств имеют как достоинства, так и недостатки. В связи с этим интересно найти такую структурную организацию управляющей логической машины, которая сочетает достоинства обоих подходов.

Пусть задан автомат $A = (X, Y, Z, \delta_y, \delta_z, y_0)$, где $X=\{x_1, \dots x_n\}$, $Z=\{z_1, \dots, z_m\}$ множество структурных (двоичных) входов и выходов; $Y=\{y_0, y_1, \dots y_s\}$ множество внутренних состояний; функции δ_y и δ_z заданы графом $G(Y, R)$, множество вершин которого есть множество состояний Y, каждое ребро $r_{ij} \in R$ взвешено функцией $f_{ij}(x_1, \dots, x_n)$ (реберной функцией), а каждая вершина взвешена набором $\tilde{z}_1, \tilde{z}_2, \dots \tilde{z}_m$ (автомат Мура).

Определим относительно разбиения $\pi = \{Y_1, Y_2, \dots Y_p\}$ множества Y такого, что $Y_i \cap Y_j = \emptyset$ для $i \neq j$, $i, j = 1, \dots, p$ и $\bigcup_{i=1}^{p} Y_i = Y$ множество автоматов $A_1, A_2, \dots, A_p$ следующим образом:

$A_i = (X_i', Y_i', Z_i', \delta_y^i, \delta_z^i, y_0^i)$ где: $X_i' = X_i \cup H_i$, $X_i \subseteq X$, $\bigcup_{i=1}^{p} X_i = X$;

$Z_i' = Z \cup Q_i$; $Y_i' = Y_i \cup \{y_0^i\}$, $Y_i \in \pi$; $\bigcup_{i=1}^{p} H_i = \bigcup_{i=1}^{p} Q_i$.

Функции δ_y^i и δ_z^i определим графом $G_i = (Y_i', R_i)$, который построим по подграфу графа G, образованному из вершин $y_j \in Y_i$ с сохранением всех внутренних ребер, этого подграфа и с замы-

канием всех "разрезанных" ребер на дополнительную вершину y_0^i. Все входящие ребра в y_0^i соответствуют "разрезанным" ребрам, ведущим в графе G из состояний блока Y_i в состояния каких-либо других блоков. Все исходящие из y_0^i ребра соответствуют "разрезанным" ребрам, ведущим в G в состояния блока Y_i из каких-либо других блоков. Взвешивание всех внутренних ребер в G_i совпадает с взвешиванием соответствующих ребер в G. "Разрезанные" ребра взвешиваются функциями:

$f_{\ell 0} = f_{\ell j} \& h_j$ для входящих ребер в y_0^i;

$f_{0j} = f_{\ell j} \& h_\ell$ для исходящих ребер из y_0^i.

При этом дополнительные входы $h_j, h_\ell \in H_i$, совпадают с дополнительными выходами q_j, q_ℓ автоматов, из или в состояния которых в G есть переходы. Входы H_i и выходы Q_i вводятся для организации взаимодействия автоматов $A_1, A_2, \ldots A_p$ Вершины $y_0^1, y_0^2, \ldots y_0^p$ графов $G_1, \ldots, G_p$ взвесим наборами $\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_n$, а все остальные вершины взвесим также как в G. Как это следует из определения δ_y^i, если A_i находится в состоянии $y_s \neq y_0^i$, то все остальные автоматы находятся в своих начальных состояниях $y_0^1, y_0^2, \ldots, y_0^p$.

Легко видеть, что автомат A и автоматы $A_1, \ldots, A_p$ (при соответствующей договоренности о вычислении выходов) реализуют одно и то же автоматное отображение.

Автоматы $A_1, \ldots, A_p$ будем называть временной (фрагментарной) декомпозицией автомата A.

Рассмотрим следующую схему вычисления (реализации) автомата A, представленного своей временной декомпозицией $A_1, \ldots, A_p$ (см.рис.2).

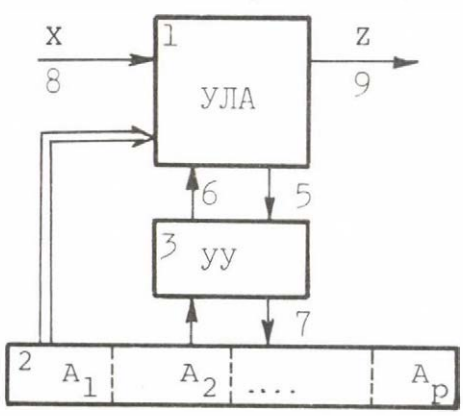


Рис. 2

Универсальный логический автомат УЛА (I) в зависимости от программирования (настройки по входам 4) может реализовать любой из автоматов $A_1, A_2, \ldots, A_P$, информация о которых хранится в разделах памяти 2. В i-том разделе памяти 2 содержится информация об автомате A_i в виде кодов настройки УЛА и адресов разделов памяти, где хранится информация об автоматах, в которые есть переход из A_i ("последователи" A_i). Опишем функционирование предложенной схемы. Пусть в УУ поступила команда на реализацию автомата A_i. УУ перекрывает каналы X и Z, считывает в УЛА соответствующую настройку по каналу 4 и запоминает адреса разделов "последователей". После этого открываются каналы X и Z и все устройство функционирует как схема жесткой логики, реализующая автомат A_i, в автоматном времени, определяемом сменой сигналов на входе X (асинхрониая реализация). Если последовательность входных сигналов приводит к переходу в начальное состояние y_0' автомата A_i, то УУ определяет в какой автомат A_j необходимо осуществить переход, перекроет каналы X и Z, считает информацию об A_j, после чего все устройство будет функционировать как схема жесткой логики, реализующая автомат A_j и т.д.

Таким образом всякий раз в структуре машины имеет место аппаратурная реализация одного из автоматов $A_1, A_2, \ldots, A_P$, а информация о перестройке и сама перестройка осуществляются программно. Реализацию устройства по описанной схеме назовём аппаратурно-программной.

4. Программируемый логический контроллер на основе аппаратурно-программного принципа реализации дискретных устройств.

Существенным в аппаратурно-программном подходе является выбор средств для реализации универсального логического автомата. В настоящей работе предлагается архитектура программируемого логического контроллера, использующего аппаратурно-программный принцип, в котором универсальный логический автомат реализуется на основе перестраиваемых сред [2]. При этом в архитектуре контроллера удалось реализовать следующие концепции:

- автоматный принцип обработки входного и формирования выходного потока, обеспечивающий малое время реакции машины;
- программный принцип настройки логического автомата, обеспечивающий высокую универсальность машины;
- фрагментарный принцип реализации алгоритма управления, заключающийся в настройке логического автомата на часть (фрагмент) алгоритма управления в соответствии с временным разбиением технологического процесса на последовательность подпроцессов;
- асинхронный принцип взаимодействия блоков машины, заключающийся в том, что новое управляющее воздействие на объект управления вырабатывается только при новом значении входных сигналов и после того, как в логическом автомате закончатся должным образом переходные процессы;
- принцип схемного описания алгоритма управления, заключающийся в представлении алгоритма управления вв виде РКС;
- принцип сочетания логических и арифметических вычислений, заключающийся в использовании результатов арифметических вычислений в логическом автомате;
- модульный принцип наращивания мощности любого блока и машины в целом.

Макроструктура предлагаемой машины изображена на рис.3, где
- программная память имеет (аналогичное назначение, что и на рис.I;
- блок УСО - устройство связи с объектом
- блок ПЛК - собственно перестраиваемый логический контроллер.

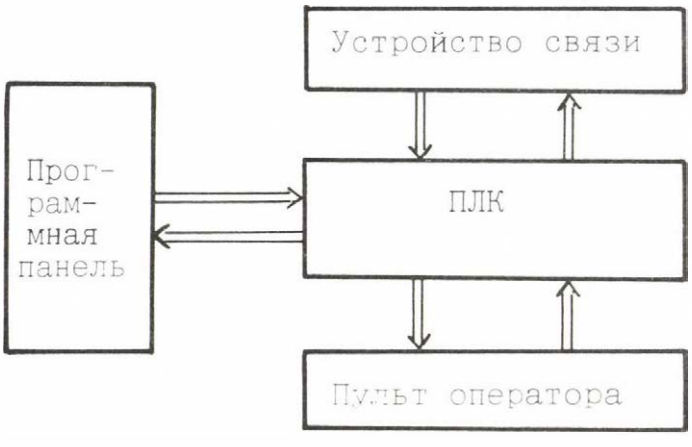


Рис. 3.

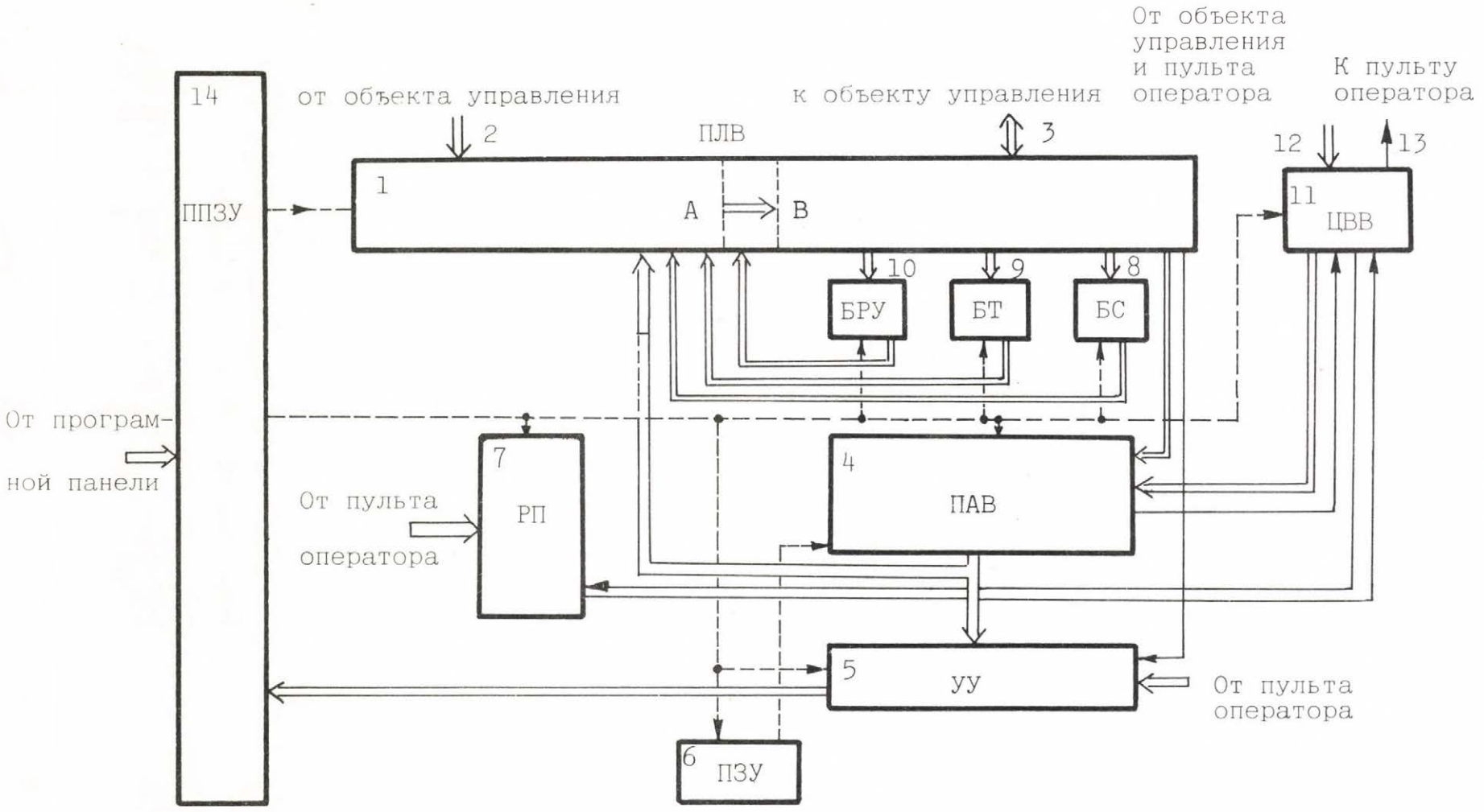

14
ППЗУ
от объекта управления
2
ПЛВ
к объекту управления
3
От объекта управления и пульта оператора
К пульту оператора
12
13
1
A
B
11
ЦВВ
10
9
8
БРУ
БТ
БС
От програм-
ной панели
7
РП
От пульта оператора
4
ПАВ
5
УУ
От пульта оператора
6
ПЗУ

Рис. 4.

Структура ПЛК изображена на рис.4. Кратко остановимся на описании отдельных блоков.

Блок ПЛВ - перестраиваемого логического вычислителя (I) представляет собой матричную однородную комбинационную структуру с переменной структурой соединений и многофункциональными ячейками и состоит из двух частей А и В. Входами А являются внешние информационные входы (2), входами В являются выходы А, а выходы В являются внешними информационными выходами (3). На любом из указанных выходах реализуется булева функция вида

$$(\ldots(1\,T_1\,a_1)\,T_2\,a_2)\,T_3\,a_3)\ldots)\,T_n\,a_n)$$

где $a_1, a_2, \ldots a_n$ являются входными переменными, T_i - логическая операция "дизъюнкция", либо "конъюнкция" либо "проникновение" (дизъюнкция с константой 0 вместо правого символа).

Блок ПЛВ предназначен для аппаратурной реализации логических вычислений над сигналами (2), поступающими от объекта, и выдачи управляющих сигналов (3) на объект. Этот блок при соответствующей настройке реализует конкретный фрагмент алгоритма управления объектом как асинхронное устройство с жесткой логикой, что обеспечивает высокое быстродействие и надежность.

Блок ПАВ - перестраиваемого арифметического вычислителя (4) представляет собой универсальную четырехнаправленную однородную вычислительную структуру с переменной структурой соединений и многофункциональными ячейками и предназначен для реализации элементов памяти (при реализации фрагментов логических вычислений с памятью) и для выполнения арифметических операций (при реализации фрагментов с арифметическими вычислениями).

Блок УУ - устройство управления (5), имеющий регулятор структуру, представляет собой совокупность регистров и предназначено для выработки адреса последующего фрагмента алгоритма управления при соответствующих воздействиях от блоков ПЛВ и ПАВ. УУ осуществляет также переключение ПЛК на обработку последующего фрагмента алгоритма управления с блокировкой информационных входов и выходов ПЛВ при настройке ПАВ и ПЛВ и других блоков ПЛК, а также аварийное прерывание и настройку блоков ПЛК на обработку фрагмента аварийного обслуживания

объекта.
Блок ПЗУ - постоянного запоминающего устройства (6) предназначен для хранения кодов настройки ПАВ на схемную реализацию арифметических операторов для реализации фрагментов с арифметическими вычислениями.
Блок РП - регистровой памяти (7) предназначен для хранения констант и исходных цифровых данных, а также для промежуточного хранения результатов обработки фрагментов с арифметическими вычислениями для их использования при обработке последующих фрагментов алгоритма управления объектом.
Блоки БС и БТ - счётчиков (8) и таймеров (9), имеющие идентичную структуру для взаимо заменяемости, предназначены для реализации временных интервалов и счёта при обработке фрагментов с логическими вычислениями.
Блок БРУ - регистров условий (I0) представляет собой набор асинхронных триггеров и предназначен для выработки логических условий при реализации фрагментов с логическими вычислениями.
Блок ЦВВ - цифрового ввода-вывода (II), представляет собой набор регистров и предназначен для последовательного ввода цифровой информации и ПАВ и РП от объекта управления и пульта оператора по внешним информационным входам (I2), а также для последовательного вывода цифровой информации из указанных блоков на пульт управления по внешнему информационному выводу (I3).
Блок памяти (I4) представляет собой ППЗУ и предназначен для хранения алгоритма управления объектом в виде совокупности фрагментов. В каждой зоне ППЗУ хранится отдельный фрагмент алгоритма управления в виде кодов настройки блоков ПЛВ и ПАВ на реализацию релейно-контактной схемы, описывающий данный фрагмент. В зоне также хранится начальная настройка блоков БС, БТ и БРУ, если они используются в реализуемом фрагменте алгоритма управления, и адреса арифметических операторов (для их извлечения из ПЗУ) и регистров РП при реализации фрагментов с арифметическими вычислениями, а также адреса фрагментов-преемников для настройки УУ.

Рассмотрим функционирование перестраиваемого логического контроллера в режимах логических и арифметических вычислений при предположении о том, что предварительно осуществлена временная декомпозиция алгоритма управления в соответствии с декомпозицией технологического процесса и соответствующие фрагменты РКС с помощью программной панели записаны в соответствующие зоны блока I4.

Функционирование ПЛК осуществляется следующим образом. С пульта оператора в выходной регистр УУ заносится адрес начального фрагмента алгоритма управления. УУ осуществляет выборку и перенос из ППЗУ настроечной информации в блоки ПЛК. Одновременно во входные регистры УУ заносятся адреса фрагментов-преемников. По окончании настройки УУ снимает блокировку информационных входов и выходов ПЛВ и переходит в состояние ожидания. При обработке фрагментов логических вычислений ПЛВ осуществляет логическую обработку внешних сигналов, поступающих от объекта управления, а также сигналов, поступающих от ПАВ,БРУ, БТ и БС, и выдачу управляющих сигналов на объект управления, а также вырабатывает сигналы запуска БС и БТ и управляет триггерами БРУ. При этом в ПАВ, в соответствии с фрагментами алгоритма управления, осуществляется переключение элементов памяти и выработка воздействий на ПЛВ. После обработки фрагмента от сигнала, поступающего из ПЛВ, запускается УУ, которое блокирует информационные входы и выходы ПЛВ. Одновременно, при помощи сигналов, поступающих от ПАВ из соответствующего входного регистра УУ осуществляется перезапись адреса последующего фрагмента в выходной регистр УУ.

При обработке фрагмента с арифметическими вычислениями по адресам, записанным в ППЗУ, осуществляется выборка из ПЗУ кодов настройки ПАВ на реализацию арифметических операторов. ПАВ осуществляет арифметические операции над последовательными двоичными числами, поступающих от ЦВВ и РП. Результаты арифметических вычислений используются в блоке ПЛВ, и могут подаваться в РП и ЦВВ.

Возможны следующие режимы функционирования ПЛК:

- логические вычисления комбинационного типа;

- арифметические вычисления;
- совмещение арифметических и логических вычислений.

ЛИТЕРАТУРА

I. Whitehouse R.A. Programmable controllers and where they are used in the industrial euviranment. Andover (Massachusetts, USA). The MODICON Corparation; 1976.

2. А.Фридман, П.Менон. Теория и проектирование переключательных схем, Изд. "Мир", Москва, 1978, стр.411-432.

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-15.1 НКС СЭВ, 1978

А. Е. ЯНКОВСКАЯ

ОПТИМИЗИРУЮЩИЕ ПРЕОБРАЗОВАНИЯ В ПРОЦЕССЕ СИНТЕЗА АСИНХРОННОГО АВТОМАТА И ИХ ПРИЛОЖЕНИЯ

Научно-исследовательский институт автоматики
и электромеханики при Томском институте
автоматизированных систем управления
и радиоэлектроники (НИИ АЭМ при ТИАСУРе)

ВВЕДЕНИЕ

Проблема создания эффективных практических алгоритмов синтеза асинхронных автоматов не нова. Но она не потеряла своей актуальности, что иллюстрируется ростом числа публикаций в научной литературе. Поскольку процедура синтеза автомата является довольно сложной, ее разбивают на последовательно выполняющиеся этапы, на каждом из которых производится оптимизация по тем или иным критериям. Одним из наиболее важных этапов является этап кодирования внутренних состояний, так как от реализации этого этапа зависит наличие или отсутствие состязаний между элементами памяти, простота структуры, надежность и т.д. Естественно, больший эффект по оптимизации структуры автомата может быть достигнут в случае, когда требования, предъявляемые к конечному результату синтеза - структуре, будут учтены на более ранних этапах синтеза. Первая попытка учета на более ранних этапах синтеза требований, предъявляемых на последующих этапах к структуре автомата, содержится в работах [I-4]. Эти работы роднит идея объединения этапов синтеза. При таком подходе сложность и трудоемкость алгоритма кодирования, на который практически ложатся функции, реализуемые несколькими этапами синтеза [I-4], в значительной степени возрастает. В связи с этим возникла заманчивая идея - произвести кодирование, обеспечивающее устойчивость автомата к опасным состязаниям между элементами памяти и, если необходимо, к отказам элементов памяти каким-либо "дешевым" способом, например, воспользовавшись наиболее простым приближенным алгоритмом [5-8], либо универсальным (типовым, то есть не зависящим от конкрет-

ных свойств матрицы переходов) [6,9] , а затем в определенной последовательности осуществить преобразование матрицы кодирования с учетом обеспечения ряда других требований: минимизации числа элементов памяти, упрощения последующих этапов синтеза, повышения надежности и т.д.

В настоящей работе ставится и решается задача оптимизирующих преобразований матрицы кодирования внутренних состояний асинхронного автомата с целью устранения избыточных элементов памяти; упрощения этапа минимизации системы булевых функций, описывающей логический преобразователь (ЛП) автомата; упрощения структуры за счет учета конкретных ограничений, связанных с используемой системой логических элементов; повышения надежности автомата за счет сокращения числа переключений элементов памяти. Приводится алгоритм устранения избыточных элементов в блоке ЛП, служащих только для возбуждения полюсов избыточных элементов памяти. Показывается применение оптимизирующих преобразований к задачам распознавания образов. Некоторые из рассматриваемых алгоритмов даны в [10]. Рамки статьи не позволяют детально изложить все алгоритмы. В связи с этим получивший наиболее широкое применение в задаче сокращения объема эксперимента при распознавании образов алгоритм устранения избыточных элементов памяти дается более подробно и сопровождается оценкой трудоемкости.
Исходной информацией для алгоритмов преобразования матрицы кодирования служит матрица переходов $\underline{\psi}$ асинхронного автомата (граф переходов, либо микропрограмма) и не полностью определенная матрица кодирования $\underline{Q}$ (строки сопоставлены внутренним состояниям, а столбцы - внутренним переменным), удовлетворяющая одному из условий устранения опасных состязаний и устойчивости автомата к t повреждениям элементов памяти:

I. Необходимое и достаточное: коды всех внутренних состояний должны находиться на расстоянии не менее h ($h = 2t + 1$), и для каждой пары переходов $a \to b$, $c \to d$, соответствующих одному и тому же входному состоянию, и таких, что $b \neq d$, в матрице $\underline{Q}$

должно найтись не менее h столбцов, в которых α-й и β-й элементы будут иметь одинаковое значение I или 0, а c-й и d-й элементы - противоположное.

2. Достаточное: коды всех внутренних состояний должны находиться на расстоянии не менее h и для каждой пары множеств K_{fg} и $K_{f\ell}$ (под множеством K_{ij} понимается совокупность внутренних состояний, из которых под воздействием входного состояния i осуществляется переход в устойчивое состояние j, также принадлежащее данной совокупности), таких, что $g \neq \ell$, в матрице $\underline{Q}$ должно найтись по крайней мере h столбцов, в которых элементы, соответствующие номерам внутренних состояний из K_{fg}, будут иметь одинаковое значение I или 0, а элементы, соответствующие номерам внутренних состояний из $K_{f\ell}$, - противоположное.

I. УСТРАНЕНИЕ ИЗБЫТОЧНОСТИ В МАТРИЦЕ КОДИРОВАНИЯ

I.I. Постановка задачи. Основные определения и понятия

Для асинхронного автомата, заданного матрицей переходов $\underline{\Psi}$ и матрицей $\underline{Q}$, удовлетворяющей условию I либо 2, требуется построить матрицу $\underline{Q}'$, удовлетворяющую условию I и состоящую из минимального подмножества столбцов матрицы $\underline{Q}$.

В дальнейшем воспользуемся следующими определениями.

Условие, заключающееся в том, что для каждой пары переходов $\alpha \to \beta$, $c \to d$ $(\beta \neq d)$, соответствующих одному и тому же входному состоянию, в матрице $\underline{Q}$ должен найтись хотя бы один столбец, α-й и β-й элементы которого будут иметь одинаковое значение I или 0, а c-й и d-й элементы - противоположное, назовем условием непересечения.

Будем говорить, что троичный вектор $\underline{\tau} = (\tau_1, \tau_2, \ldots, \tau_n)$, где n - число внутренних состояний, задает условие непересечения для пары переходов $\alpha \to \beta$, $c \to d$ $(\beta \neq d)$, если компоненты $\tau_\alpha = \tau_\beta$; $\tau_c = \tau_d$, а остальные компоненты принимают значение "-" ("-" - символ неопределенности).

Запись $\tau_i \Leftarrow \underline{q}_j$ означает, что условие $\underline{\tau}_i$ обеспечивается (импли-

цируется) [II] j -м столбцом матрицы $\underline{Q}$. Например, при $\underline{z}_i$ = I0-0-0, $\underline{q}_j$ = I0I0-0, имеет место I0-0— $\Leftarrow$ I0I0-0 и I0-0—$\Leftarrow$ 0I0I-I. Для выполнения условий I, 2 необходимо, чтобы полученный для каждой пары существенных переходов вектор $\underline{z}_i$ имплицировался не менее h столбцами матрицы $\underline{Q}$. Переход вида $a \to a$ при наличии перехода $\beta \to a$ при заданном входном состоянии назовем несущественным. Остальные переходы будем называть существенными.

И, наконец, введем следующие обозначения: U - множество строк матрицы $\underline{U}$, $|U|$ - число строк матрицы $\underline{U}$, m - число столбцов матрицы $\underline{Q}$, $]\log_2 n[$ - наименьшее сверху целое к $\log_2 n$.

I.2. Алгоритм

Суть алгоритма сводится к выделению минимального подмножества столбцов матрицы $\underline{Q}$, которые в совокупности имплицируют (обеспечивают) не менее h раз все условия непересечения, порождаемые матрицей переходов $\underline{\Psi}$. Алгоритм построения матрицы $\underline{Q}'$ состоит из следующих процедур.

I. Построить булеву матрицу импликаций $\underline{U}$, строки которой сопоставлены условиям непересечения, а столбцы внутренним переменным, причем u_{ij} = I $\leftrightarrow$ $\underline{z}_i \Leftarrow \underline{q}_j$.
2. Удалить в матрице $\underline{U}$ поглощающие строки (строка $\underline{u}_i$ поглощает строку $\underline{u}_j$, если $\underline{u}_i \wedge \underline{u}_j = \underline{u}_j$ и $\bar{\underline{u}}_i \wedge \underline{u}_j = \phi$), то есть построить безызбыточную матрицу импликаций, обозначаемую далее через $\underline{U}'$.
3. Найти кратчайшее h - кратное столбцовое покрытие матрицы $\underline{U}'$, то есть минимальное по мощности подмножество столбцов, содержащих в совокупности не менее h единиц в каждой строке.
4. Построить матрицу $\underline{Q}'$ из тех столбцов матрицы $\underline{Q}$, которые сопоставлены столбцам матрицы $\underline{U}'$, вошедшим в кратчайшее h - кратное столбцовое покрытие.

В связи с тем, что число условий непересечения, равное числу строк матрицы $\underline{U}$, может быть довольно большим и достигать величины $3C_n^4$ [9], предлагается алгоритм непосредственного по-

строения из матриц $\underline{\Psi}$, $\underline{Q}$ матрицы $\underline{U}'$.

I. Дополнить матрицу $\underline{\Psi}$ столбцом устойчивых состояний [5], что необходимо для различения кодов внутренних состояний.
2. Удалить из матрицы $\underline{\Psi}$ несущественные переходы.
3. Выделить согласно условию I очередную пару существенных переходов, построить для нее условие непересечения $\underline{z}$ и соответствующую строку матрицы $\underline{U}^P$ (через $\underline{U}^P$ обозначено промежуточное значение матрицы $\underline{U}'$, полученное в процессе ее построения). Удалить из матрицы $\underline{U}^P$ поглощающие строки.
4. Проверить наличие нерассмотренных согласно условию I пар существенных переходов. При наличии - переход к п. 3, иначе $\underline{U}' := \underline{U}^P$ (запись $B := A$ означает присвоение величине B значения A).

С целью сокращения перебора при поиске кратчайшего h - кратного столбцового покрытия матрицы $\underline{U}'$ предлагаются оптимизирующие процедуры, связанные со свойствами матрицы $\underline{U}'$:
α) столбцы матрицы $\underline{U}'$, сопоставленные единичным элементам строки, содержащей ровно h единичных элементов, входят в ядро кратчайшего h - кратного покрытия;
β) если дизъюнкция всех строк матрицы $\underline{U}'$, каждая из которых содержит ровно h единичных элементов, равна m - компонентному вектору, компоненты которого суть I, то матрица $\underline{Q}$ является безызбыточной.

В заключении заметим, что алгоритм [II] устранения избыточности в матрице кодирования, обеспечивающей только устранение опасных состязаний между элементами памяти ($t = 0$), является частным случаем вышеописанного алгоритма построения матрицы $\underline{Q}$.

I.3. Пример

Проиллюстрируем алгоритм на следующем примере: $t = I$, $h = 3$,

$$\underline{\Psi} = \begin{bmatrix} I & I & I & 5 \\ I & 2 & 2 & 2 \\ 3 & 2 & 4 & 5 \\ 3 & 2 & 4 & 5 \\ - & 5 & 4 & 5 \end{bmatrix} \begin{matrix} I \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \qquad \underline{Q} = \begin{bmatrix} I & I & I & I & I & I & 0 & 0 & 0 & 0 & I & 0 \\ I & I & I & 0 & 0 & 0 & I & I & I & I & 0 & I \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & I & I \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & I & I & 0 & 0 & 0 & 0 & I & I \end{bmatrix} \begin{matrix} I \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} .$$

Отметив, что строки матрицы $\underline{\Psi}$ сопоставлены внутренним состояниям, а столбцы – входным состояниям, перейдем к построению матрицы $\underline{U}'$. В результате реализации первых двух пунктов алгоритма получим скорректированную матрицу $\underline{\Psi}$:

$$\underline{\Psi} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{bmatrix} - & 1 & 1 & 5 & 1 \\ 1 & - & 2 & 2 & 2 \\ - & 2 & 4 & 5 & 3 \\ 3 & 2 & - & 5 & 4 \\ - & 5 & 4 & - & 5 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$

Выделим в ней первую пару существенных переходов $2 \to 1$, $4 \to 3$, построим вектор $\underline{z}$ и матрицу $\underline{U}^P$: $\underline{z}$ = 1100–, $\underline{U}^P$ = [111000000000]. Перейдем к п. 4, а затем к п. 3. Выделим пару $1 \to 1$, $3 \to 2$, построим вектор $\underline{z}$ и матрицу $\underline{U}^P$: $\underline{z}$ = 100––,

$$\underline{U}^P = \begin{array}{c} \begin{matrix} & & & & & & & & & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 & 2 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

Поскольку алгоритм построения матрицы $\underline{U}'$ прост, а иллюстрация его на каждом шаге громоздка, прервем ее и приведем матрицу $\underline{U}'$:

$$\underline{U}' = \begin{array}{c} \begin{matrix} & & & & & & & & & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 0 & 1 & 2 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} 2 \to 1,\ 3 \to 4 \\ 1 \to 1,\ 3 \to 2 \\ 1 \to 1,\ 4 \to 2 \\ 3 \to 3,\ 4 \to 4 \end{matrix}$$

Справа от матрицы $\underline{U}'$ приведены пары существенных переходов, порождающие соответствующие строки матрицы $\underline{U}'$. Столбцы 1–5, 10–12 матрицы $\underline{U}'$ составляют кратчайшее трехкратное покрытие, а столбцы 1–3, 10–12 – ядро кратчайшего трехкратного покрытия этой матрицы (см. процедуру α). Матрица $\underline{Q}'$ принимает вид:

$$\underline{Q}' = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$

Заметим, что при t = 0, h = 1 в результате применения данного алгоритма к приведенному примеру получим следующую матрицу $\underline{Q}'$:

$$\underline{Q}' = \begin{array}{c} \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$

I.4. Трудоемкость алгоритма

Из всех процедур, реализуемых алгоритмом, наиболее трудоемкой, особенно при больших размерах матрицы $\underline{U}'$, является процедура поиска кратчайшего h - кратного покрытия матрицы $\underline{U}'$. В связи с этим приведем оценку числа строк этой матрицы.

Теорема. $|U'| = C_m^{\left]\frac{m}{\log_2 n}\right[}$.

Доказательство. Поскольку в матрице $\underline{U}'$ отсутствуют поглощающие строки, то очевидно, что $|U'|$ максимально при равновероятном распределении в матрице $\underline{U}'$ единичных элементов. Зная среднее число единичных элементов в строках матрицы $\underline{U}'$, можно оценить число строк этой матрицы. Для того, чтобы каждое условие непересечения обеспечивалось хотя бы одним столбцом матрицы $\underline{Q}$, необходимо иметь длину кода внутренних состояний, не меньшую $\log_2 n$ (оценку $]\log_2 n[$ не используем, поскольку некоторые элементы матрицы $\underline{Q}$ могут быть неопределены). Так как в результате кодирования получена длина кода, равная m, то каждое из условий непересечения $\underline{r}$ в среднем имплицируется $\frac{m}{\log_2 n}$ раз, то есть среднее число единичных элементов в строке матрицы $\underline{U}'$ не превышает $\frac{m}{\log_2 n}$. Отсюда $|U'| \le C_m^{\left]\frac{m}{\log_2 n}\right[}$, что и требовалось доказать.

Поскольку число элементов памяти при кодировании внутренних состояний для практически важного класса автоматов приближенными алгоритмами при $t = 0$ не превышало, как правило, $2\log_2 n$, то $|U'| \le C_m^2$. На практике ввиду неравномерности распределения единичных элементов в матрице $\underline{U}'$, число ее строк значительно меньше полученной оценки и близко к m', где m' - число столбцов матрицы $\underline{Q}'$.

I.5. Апробация алгоритма на ЭВМ

Описанный алгоритм выражен в языке ЛЯПАС [I4] и опробован на ЭВМ М-222, в основном на реальных примерах. В качестве исходной матрицы кодирования служили матрицы, полученные при $t = 0$ с помощью шагово-циклического алгоритма [5], и несколько универсальных матриц [9]. Число $n \leq 64$. Время работы алгоритма не превышало 3-х минут. Число $m - m'$ не превышало двух для матриц кодирования, построенных по алгоритму [5] и достигало величины $s_n - \log_2 n$ для универсальных матриц кодирования, где s_n - длина универсального кода. Программа рассчитана на $n \leq 5II$.

2. УСТРАНЕНИЕ ИЗБЫТОЧНОСТИ В БЛОКЕ ЛОГИЧЕСКОГО ПРЕОБРАЗОВАТЕЛЯ

При устранении избыточных элементов памяти могут появиться избыточные элементы в блоке ЛП, то есть те элементы, которые служат только для возбуждения полюсов избыточных элементов памяти. Исходной информацией для алгоритма устранения избыточных элементов в блоке ЛП служит граф соединения функциональных элементов автомата с указанием вершин, сопоставленных избыточным элементам памяти. Суть алгоритма заключается в следующем.

По графу соединения функциональных элементов строится булева матрица $\underline{v}$, строки которой сопоставлены функциональным элементам автомата за исключением элементов памяти, а столбцы - элементам памяти (столбцам матрицы $\underline{Q}'$) и выходным полюсам автомата. Если функциональный элемент p_i соединен непосредственно или через ряд других элементов с элементом памяти z_j (элемент z_j сопоставлен внутренней переменной q_j) или с выходным полюсом y_j, то элемент v_{ij} матрицы $\underline{v}$ принимает единичное значение, в противном случае - нулевое. Нулевые строки матрицы $\underline{v}$ сопоставлены искомым избыточным элементам ЛП.

3. ВЫДЕЛЕНИЕ СУЩЕСТВЕННЫХ ВНУТРЕННИХ ПЕРЕМЕННЫХ

3.I. Постановка задачи

Задача выделения существенных внутренних переменных представляет интерес в связи с возможностью облегчения процесса синтеза на последующих этапах, в основном на этапе минимиза-

ции системы булевых функций в классе ДНФ. С целью постановки данной задачи воспользуемся введенными ранее определениями, а также следующими:

При заданном входном состоянии f совокупность условий непересечения для всех пар переходов, содержащих переход q^f (q-й переход при f-м входном состоянии), назовем условием непересечения для этого перехода.
Существенными внутренними переменными для перехода q^f назовем такие, которые соответствуют столбцам матрицы $\underline{Q}$, обеспечивающим в совокупности не менее h раз условие непересечения для перехода q^f, причем эта совокупность внутренних переменных должна обладать для заданной матрицы $\underline{U}$ минимальной мощностью.

Задача выделения существенных внутренних переменных в матрице формулируется так:

для асинхронного автомата, заданного матрицей переходов $\underline{Y}$ и матрицей $\underline{Q}$, удовлетворяющей условию I либо 2, требуется построить двоичную матрицу $\underline{E}$, представляющую набор существенных внутренних переменных для каждого существенного перехода.

3.2. Алгоритм

Для изложения алгоритма введем следующие обозначения и определения.

Совокупность существенных внутренних переменных для существенного перехода q^f обозначим e_{q^f}, а строку матрицы $\underline{E}$, сопоставленную переходу q^f, - $\underline{e}_{q^f}$. Через m^f обозначим число существенных переходов при f-м входном состоянии, через p_{q^f} - число всех пар существенных переходов, связанных условием непересечения и содержащих переход q^f, а через k - число входных состояний автомата.

Частичной матрицей импликаций $\underline{U}^{q^f}$ назовем такую булеву матрицу, строки которой сопоставлены условиям непересечения для всех

пар существенных переходов, содержащих переход g^t; столбцы - столбцам матрицы $\underline{Q}$ (внутренним переменным), а элемент $u_{ij}^{g^t} = I \rightarrowtail \underline{r}_i^{g^t} \Leftarrow \underline{q}_j$, где $\underline{r}_i^{g^t}$ - условие непересечения для i -й пары существенных переходов, содержащих переход g^t. Термин частичная отражает тот факт, что строки матрицы порождаются только условием непересечения для перехода g^t.

Алгоритм построения матрицы $\underline{E}$ состоит из следующих процедур:

I. $E^- := \phi$, $f := I$.
2. $g := I$
3. Построить матрицу $\underline{U}^{g^t}$. Найти ее кратчайшее h - кратное столбцовое покрытие. Получить $\underline{e}_{g^t}$, $E^- := E^- \vee \underline{e}_{g^t}$, $g := g + I$. Если $g > m^t$, то перейти к п. 4, иначе - к п. 3.
4. $f := f + I$. Если $f \le k$, то перейти к п. 2, иначе $E := E^-$ и построить матрицу $\underline{E}$.

Данный алгоритм имеет много общего с алгоритмом, приведенным в подразделе I.2. Задача поиска кратчайшего h - кратного столбцового покрытия вычислительных трудностей, связанных с временными затратами, не представляет, поскольку $|U^{g^t}| \le p_{g^t} < n$. Повысить быстродействие приведенного алгоритма можно за счет введения операции удаления поглощающих строк в процессе построения матрицы $\underline{U}^{g^t}$.

И, наконец, отметим, что в качестве исходной матрицы кодирования для данного алгоритма целесообразно применять матрицу $\underline{Q}'$.

3.3. Пример

Поскольку алгоритм довольно прост и является циклическим, проиллюстрируем его на примере, приведенном в подразделе I.3 (см. матрицу $\underline{\Psi}$ и матрицу Q', полученную при $t = I$), построив только одну строку $\underline{e}_4{}^3$, соответствующую 4-у существенному переходу 5→4 при 3-м входном состоянии. В результате реализации алгоритма матрица $\underline{U}^{4^3}$ и строка $\underline{e}_4{}^3$ примут следующий вид:

$$\underline{U}^{4^3} = \begin{bmatrix} I\ I\ I\ 0\ 0\ 0\ 0\ 0 \\ I\ I\ I\ 0\ 0\ I\ 0\ 0 \end{bmatrix} \begin{matrix} 5 \rightarrow 4,\ I \rightarrow I \\ 5 \rightarrow 4,\ 2 \rightarrow 2 \end{matrix}, \quad \underline{e}_4{}^3 = \text{III00000}.$$

4. ПРЕОБРАЗОВАНИЕ МАТРИЦЫ КОДИРОВАНИЯ В ЦЕЛЯХ УЧЕТА ПАРАМЕТРОВ ЛОГИЧЕСКИХ ЭЛЕМЕНТОВ

Учет числа входов логических элементов можно произвести на этапе кодирования внутренних состояний путем регулирования числа существенных внутренних переменных для каждого существенного перехода [I3]. Кроме того, уже построенную матрицу можно преобразовать в целях учета числа входов логических элементов. Данная задача формулируется следующим образом:

для асинхронного автомата, заданного матрицей переходов $\underline{\psi}$ и матрицей $\underline{Q}$, удовлетворяющей условию I либо 2, требуется преобразовать матрицу $\underline{Q}$ по возможности с меньшим увеличением числа ее столбцов, таким образом, чтобы она удовлетворяла условию I и число существенных внутренних переменных для каждого существенного перехода не превышало заданного.

Кроме учета числа входов логических элементов на этапе кодирования следует рассмотреть и задачу учета нагрузочной способности логических элементов. "Нагрузка" на элементы, реализующие ДНФ функции возбуждения и выходов в большинстве случаев на практике не превышает нагрузочную способность этих элементов [I5]. Превышение "нагрузки" над нагрузочной способностью элементов наблюдалось, как правило, во многих практических случаях на выходных полюсах элементов памяти. Задача учета этой "нагрузки" при кодировании внутренних состояний асинхронного автомата сформулирована в [I5]. С нашей точки зрения, представляет интерес и нижеформулируемая задача перераспределения "нагрузки" на выходных полюсах элементов памяти:

для асинхронного автомата, заданного матрицами переходов и выходов и матрицей $\underline{Q}$, удовлетворяющей условию I либо 2, произвести преобразование матрицы $\underline{Q}$ по возможности с меньшим увеличением числа ее столбцов таким образом, чтобы она удовлетворяла условию I и каждая существенная внутренняя переменная принимала значение I(0) не более, чем на q -I существенных переходах, где q -нагрузочная способность элементов памяти.

В предположении, что блок памяти автомата реализуется на триггерах с двумя выходными полюсами и петли обратной связи уменьшают нагрузочную способность триггера на единицу, метод решения поставленной задачи сводится к построению двух булевых матриц, аналогичных матрице импликаций $\underline{U}$, и преобразованию их к такому виду, чтобы число единичных элементов в каждом столбце не превышало заданного (g - I) при минимальном увеличении числа их столбцов (столбцов матрицы $\underline{Q}$). При этом преобразованная матрица кодирования должна удовлетворять условию I.

5. ДООПРЕДЕЛЕНИЕ МАТРИЦЫ КОДИРОВАНИЯ

При построении и вышеизложенных преобразованиях матрица кодирования $\underline{Q}$ получается, как правило, не полностью определенной. От структуры матриц $\underline{\Psi}$, $\underline{Q}$ зависит число переключений элементов памяти, являющееся показателем надежностной работы автомата. Сокращение числа переключений достигается минимизацией функции $W=\sum_{\alpha=1}^{n-1}\sum_{\beta=\alpha+1}^{n} P_{\alpha\beta}\,\ell_{\alpha\beta}$, где $P_{\alpha\beta}$ - число переходов между состояниями α и β ; $\ell_{\alpha\beta}$ - расстояние по Хэммингу между кодами этих состояний. Функция W была введена Армстронгом [I6] для кодирования состояний синхронных автоматов с учетом упрощения структуры ЛП.

Минимизация функции W может быть достигнута путем субоптимального доопределения не полностью определенной матрицы $\underline{Q}$ до двоичной. Алгоритм, результаты статистического эксперимента на ЭВМ и иллюстрирующий пример приведены в [I2]. Число переключений элементов памяти при применении алгоритма сокращалось до I4 % по сравнению с доопределением неопределенных элементов матрицы $\underline{Q}$ нулевыми значениями.

6. ПРИМЕНЕНИЕ ОПТИМИЗИРУЮЩИХ ПРЕОБРАЗОВАНИЙ В РАСПОЗНАВАНИИ ОБРАЗОВ

Большое внимание, уделяемое в последнее время задачам распознавания образов, не случайно. Как справедливо указано в [I7], распознавание образов стало важным инструментом исследований в управлении, проектировании, геологии, медицине, социологии, искусственном интеллекте и других. Рассматриваемая в настоя-

щей работе задача сокращения объема эксперимента при распознавании образов близка по постановке к задаче минимизации числа признаков в распознающих системах [I7]. В отличие от [I7] состояние признака в данной работе может принимать значение из множества $\{0,1,-\}$, где "-" символ неопределенности (невозможно измерить наличие или отсутствие признака, или данный признак несущественен для рассматриваемого объекта). Кроме того, если в [I7] ставится задача попарной различимости описаний объектов, то в нашем случае дополнительно рассматривается и задача различимости непересекающихся классов объектов $S_{f1}, S_{f2}, \ldots, S_{fp_f}$ для каждой заданной совокупности объектов $S_f \subseteq S (f \in \{1,2,\ldots,k\})$. Таким образом, сокращение объема эксперимента при распознавании достигается путем устранения избыточных признаков с точки зрения распознаваемости объектов и классов объектов для каждой совокупности S_f в предъявляемых описаниях. Исходными данными для алгоритма устранения избыточных признаков являются описания объектов, представленные аналогично матрице кодирования внутренних состояний асинхронного автомата троичной матрицей $\underline{Q}$, строки которой сопоставлены объектам, столбцы - признакам, элемент $q_{ij} = 1(0)$ тогда и только тогда, когда объекту $s_i \in S$ ($i \in \{1,2\ldots,n\}$) присущ (не присущ) признак q_j и элемент $q_{ij} = -$, когда значение признака q_j для объекта s_i неопределено. Дополнительно должна быть задана информация о том, какие описания объектов и классов объектов подлежат различению. Эту информацию удобно представить матрицей $\underline{\Psi}$, которая аналогична матрице переходов асинхронного автомата. Элемент ψ_{if} принимает значение номера первого элемента из класса совокупности S_f, элементом которого и является объект s_i (аналогия с K - множествами). Если различению подлежат описания объектов, то к матрице $\underline{\Psi}$ добавляется столбец, подобный столбцу устойчивых состояний [5].

Таким образом, рассматриваемая в данном разделе задача сводится к задаче устранения избыточности в матрице кодирования внутренних состояний асинхронного автомата. Сокращение объема эксперимента при различении одного класса объектов от других из заданной совокупности может быть достигнуто с помощью ал-

горитма выделения существенных внутренних переменных. Представляет интерес и задача доопределения описаний в случае неоднозначности распознавания самих объектов либо их классов. Алгоритм решения сводится к достраиванию текущей матрицы кодирования [5]. Следовательно, автоматные модели и логико-комбинаторные методы (в частности, оптимизирующие преобразования), используемые при кодировании внутренних состояний асинхронного автомата, применимы и к задачам распознавания образов.

Использование алгоритмов оптимизирующих преобразований для оптимального выбора количества параметров при исследовании петрофизических свойств горных пород в условиях, близких к их естественному залеганию, позволяет при меньших затратах значительно ускорить эксперимент.

Л И Т Е Р А Т У Р А

I. Балаклей Л. И., Янковская А. Е. Объединение этапов синтеза асинхронного автомата. Сб. "Тезисы докладов П Всесоюзного совещания по теории релейных устройств и конечных автоматов", Рига, "Зинатне", 1971.

2. Frosini G. Influence of State Reduction on the Number of State Variables in Race-Free Asynchronous Sequential Circuits. "Information and Control", 1972, v.20, No.1.

3. Халлбауэр Г. Процедуры минимизации и размещения состояний при синтезе асинхронных последовательностных схем, выполняемых за один шаг. Сб."Дискретные системы,т.I", "Зинатне",1974.

4. Янковская А. Е. Алгоритм субоптимального сквозного логического синтеза асинхронных автоматов, ориентированный на машинную реализацию. Сб. "Теория релейных устройств. Труды XУI Всесоюзной школы-семинара", Челябинск, ЧПИ, 1976.

5. Янковская А. Е. Алгоритмы кодирования внутренних состояний асинхронных автоматов. Сб. "Цифровые модели и интегрирующие структуры", Таганрог, ТРТИ, 1970.

6. Сагалович Ю. Л. Кодирование состояний и надежность автоматов, М., "Связь", 1975.

7. Гаврилов М. А., Остиану В. М., Потехин А. И. Надежность дискретных систем. Сб. "Теория вероятностей. Мат. статистика. Теор. кибернетика. 1969. (Итоги науки, ВИНИТИ АН СССР)", М., 1970.

8. Потехин А. И., Рогинский В. Н. Динамика дискретных автоматов. Сб. "Теория вероятностей. Мат. статистика. Теор. кибернетика, т. 14. (Итоги науки. ВИНИТИ АН СССР)", М., 1977.

9. Friedman A. D., Graham R. L., Ullman J. D. Universal single transition time asyncronous state assignments. "IEEE Trans. on Computers", 1969, v. C-18, No.6.

10. Янковская А. Е. Оптимизирующие преобразования в процессе кодирования внутренних состояний асинхронного автомата. Сб. "УП Всесоюзное совещание по проблемам управления, книга 2, тезисы докладов", М., Минск, 1977.

11. Янковская А. Е. К задаче устранения избыточности в матрице кодирования внутренних состояний асинхронного автомата. Сб. "Автоматизация обработки информации", Томск, ТГУ, 1977.

12. Янковская А. Е. Алгоритм сокращения числа переключений элементов памяти асинхронного автомата. Там же.

13. Синтез асинхронных автоматов на ЭВМ. Авторский коллектив под общей редакцией Закревского А. Д., Минск, "Наука и техника", 1975.

14. Закревский А. Д. Алгоритм синтеза дискретных автоматов, М., "Наука", 1971.

15. Денисенко Е. М. О некоторых ограничениях, накладываемых на кодирование внутренних состояний многотактных управляющих устройств. - "Кибернетика", 1969, № 5.

16. Armstrong D. B. A programmed algorithm for assigning internal codes for sequental machines."IRE Trans", 1962, EC - 11, No.4.

17. Карелина А. В., Печерский Ю. Н. Теоретико-графические методы в распознавании образов, Кишинев, "Штиинца", 1978.

MTA SZTAKI TANULMÁNYOK 99/1979. - Симпозиум по теме 1-16.1 НКС СЭВ, 1978

БАРАНОВ С.И., СИНЕВ В.Н.

МИКРОПРОГРАММНЫЕ АВТОМАТЫ И ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ МАТРИЦЫ

Ленинградский институт точной механики и оптики

ВВЕДЕНИЕ

В работе [1] рассмотрены вопросы синтеза микропрограммных автоматов (МПА) на матрицах. Вместе с тем, в настоящее время серийно выпускаются стандартные большие интегральные схемы с матричной структурой, так называемые программируемые логические матрицы (ПЛМ) [2]. В данной работе в развитие результатов, полученных в [1], представлены методы структурного синтеза МПА на стандартных ПЛМ.

1. ПРОГРАММИРУЕМЫЕ ЛОГИЧЕСКИЕ МАТРИЦЫ

В зависимости от структурной организации ПЛМ можно разделить на ПЛМ комбинационной логики и ПЛМ с памятью.

ПЛМ комбинационной логики (рис. 1,а) состоит из двух матриц M_1 и M_2, первая из которых формирует q конъюнктивных термов от s входных переменных (или их отрицаний), а вторая - t дизъюнкций от термов, полученных в матрице M_1. Таким образом, основными параметрами ПЛМ комбинационной логики являются число входов s, число выходов t и число термов q. ПЛМ (s,t,q) - комбинационная схема, и любая система $y_1,\dots y_N$ $(N \leqslant t)$ булевых функций от $x_1,\dots,x_L$ $(L \leqslant s)$ переменных, дизъюнктивная нормальная форма (ДНФ) которых содержит H термов $(H \leqslant q)$, может быть реализована на одной ПЛМ. Условное изображение ПЛМ (s,t,q) приведено на рис. 1,б. На рис. 2 иллюстрируется реализация на ПЛМ(3,3,6) системы булевых функций:

$$y_1 = x_1\bar{x}_2 \vee \bar{x}_1x_2 \vee \bar{x}_1\bar{x}_3 ;$$
$$y_2 = x_1\bar{x}_2 \vee x_1x_3 ;$$
$$y_3 = \bar{x}_1x_3 \vee \bar{x}_1x_2 \vee x_1x_2\bar{x}_3 .$$

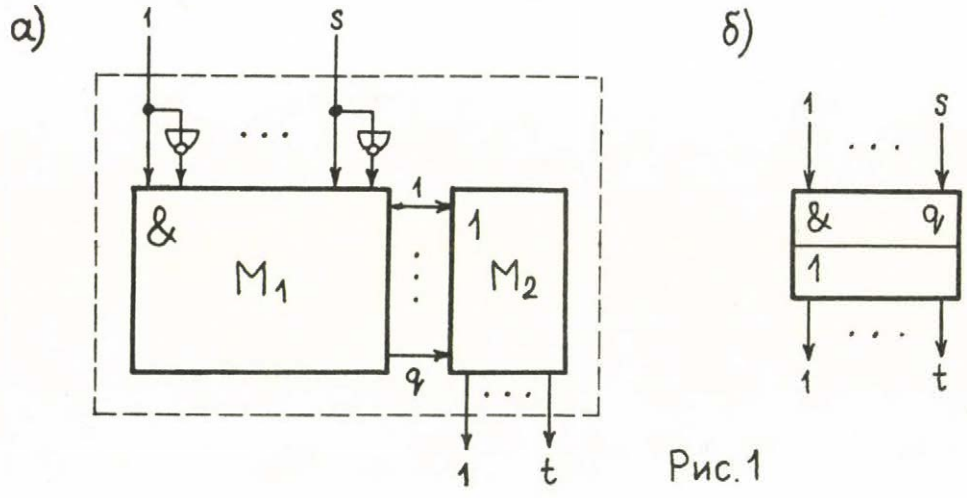


Рис.1

Выпускаемые в настоящее время ПЛМ (s,t,q) имеют следующие значения параметров: $12 \leq s \approx t \leq 20$, $48 \leq q \leq 96$.

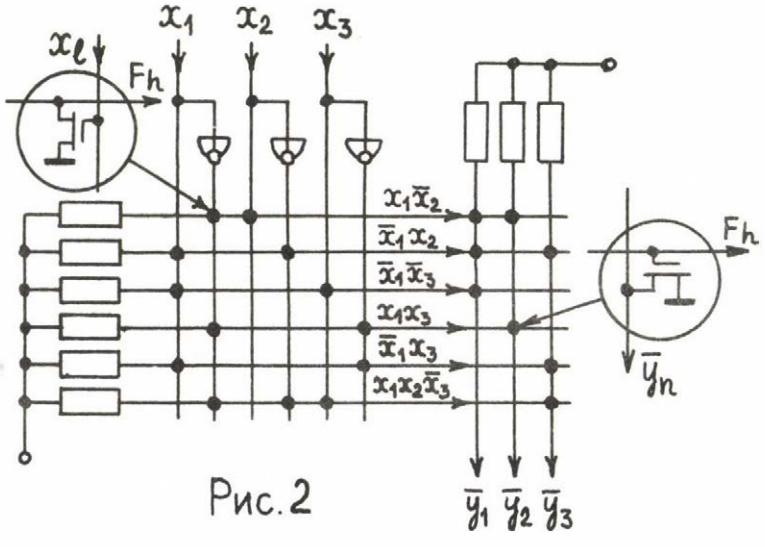


Рис.2

В отличие от ПЛМ (s,t,q) ПЛМ с памятью (рис. 3,а) содержит r **- разрядный регистр RG в цепи обратной связи между матрицами** M_2 **и** M_1**, вследствие чего матрица** M_1 **формирует** q **термов от** $s+r$ **переменных (или их отрицаний), а матрица** M_2 **реализует** $t+r$ **дизъюнкций от термов, полученных в матрице** M_1**. Условное изображение ПЛМ (s,t,q,r) приведено на рис. 3,б. ПЛМ (s,t,q,r) можно применять вместо ПЛМ (s,t,q) , но основное её назначение - реализация устройств управления.**

В зависимости от способа организации межсоединений шин в

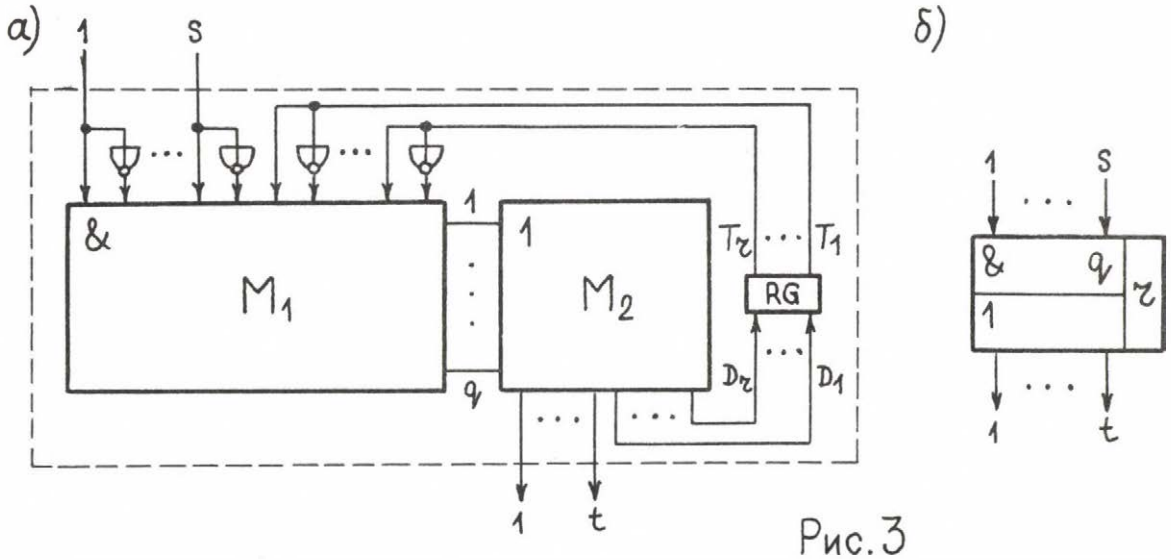


Рис.3

матрицах (способа программирования) различают ПЛМ, программируемые в процессе изготовления, и ПЛМ, программируемые пользователем. Последние существуют как однократно программируемые, так и репрограммируемые [3].

2. СИНТЕЗ МИКРОПРОГРАММНЫХ АВТОМАТОВ НА ПЛМ (S,t,q)

2.1. Тривиальная реализация

Под МПА **S** будем понимать пятерку $S=(A,X,Y,\delta,\lambda)$, в которой $A=\{a_1,\dots,a_M\}$ – множество состояний, $X=\{x_1,\dots,x_L\}$ и $Y=\{y_1,\dots,y_N\}$ – множества входных и выходных каналов (двоичных переменных), δ и λ – функции переходов и выходов соответственно. При структурном синтезе МПА принято задавать [4] в виде таблиц переходов с пятью столбцами: a_m – исходное состояние, a_s – следующее состояние, X_h – входной сигнал на переходе (a_m,a_s), Y_t – выходной сигнал на переходе (a_m,a_s), $h=1,\dots,H$ – номер перехода. Пример задания МПА приведен в табл. 1. Входные сигналы МПА **S** – элементы множества $D=\{d_1,\dots,d_{2^L}\}$, где $d_i=(e_{i1},\dots,e_{iL})$ – L – компонентный вектор $(e_{il}\in\{0,1\})$. Выходные сигналы МПА **S** – элементы множества $G=\{g_1,\dots,g_{2^N}\}$, где $g_j=(e_{j1},\dots,e_{jN})$ – N – компонентный вектор $(e_{jn}\in\{0,1\})$. Функции переходов и выходов реализуют отображения $\delta: A\times D\to A$ и $\lambda: A\times D\to G$. В таблице переходов МПА строка $a_m a_s X_h Y_t$ означает, что МПА пе-

Таблица 1

a_m	a_s	X_h	Y_t	h
a_1	a_3	x_2x_3	Y_4 $y_1y_2y_4$	1
	a_1	$x_2\bar{x}_3$	Y_0 —	2
	a_2	$\bar{x}_2$	Y_1 y_4	3
a_2	a_5	x_1	Y_7 y_8y_9	4
	a_3	$\bar{x}_1$	Y_9 y_9	5
a_3	a_4	x_3	Y_6 y_5	6
	a_3	$\bar{x}_3\bar{x}_6$	Y_5 y_2y_4	7
	a_4	$\bar{x}_3x_6$	Y_8 y_7y_{10}	8
a_4	a_1	1	Y_6 y_5	9
a_5	a_5	x_1	Y_2 y_3y_6	10
	a_1	$\bar{x}_1x_4x_5$	Y_0 —	11
	a_2	$\bar{x}_1\bar{x}_4x_5$	Y_7 y_8y_9	12
	a_5	$\bar{x}_1\bar{x}_5$	Y_3 y_8	13

Таблица 2

a_m	a_s	X_h	Y_t	h
a_1	a_3	x_2x_3	Y_4 $y_1y_2y_4$	1
	a_1	$x_2\bar{x}_3$	Y_0 —	2
	a_2	$\bar{x}_2$	Y_1 y_4	3
a_3	a_4	x_3	Y_6 y_5	6
	a_3	$\bar{x}_3\bar{x}_6$	Y_5 y_2y_4	7
	a_4	$\bar{x}_3x_6$	Y_8 y_7y_{10}	8
a_4	a_1	1	Y_6 y_5	9
a_2	a_5	x_1	Y_7 y_8y_9	4
	a_3	$\bar{x}_1$	Y_9 y_9	5
a_5	a_5	x_1	Y_2 y_3y_6	10
	a_1	$\bar{x}_1x_4x_5$	Y_0 —	11
	a_2	$\bar{x}_1\bar{x}_4x_5$	Y_7 y_8y_9	12
	a_5	$\bar{x}_1\bar{x}_5$	Y_3 y_8	13

реходит из состояния a_m в a_s под действием всех тех входных наборов из множества D, на которых конъюнкция $X_h=1$. При этом на выходе автомата выдается выходной набор $g_t \in G$, у которого компоненты, входящие в выходной сигнал (микрокоманду) Y_t, равны единице, а остальные - нулю.

Метод синтеза МПА S на ПЛМ (s,t,q) определяется параметрами L, N, H, R автомата S и ПЛМ (s,t,q), где L и N - числа входных и выходных переменных, H - число переходов, $R =]\log_2 M[$ - разрядность кода $K(a_m) = (e_{m1}, \ldots, e_{mR})$ состояния a_m $(m=1,\ldots,M)$, $]\beta[$ означает ближайшее целое число, большее β или равное ему, если β целое.

Ясно, что если $L+R \le s$, $N+R \le t$ и $H \le q$, то МПА S

реализуется тривиальным образом на одной ПЛМ (s,t,q) в виде структуры, изображенной на рис. 4 (предполагается, что в качестве элементов памяти используются D - триггеры). Так,

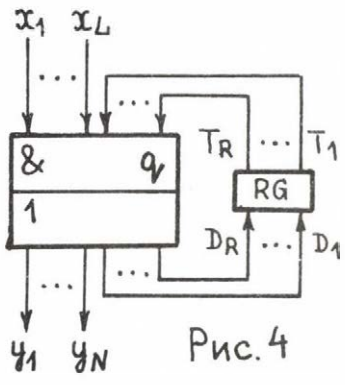


Рис. 4

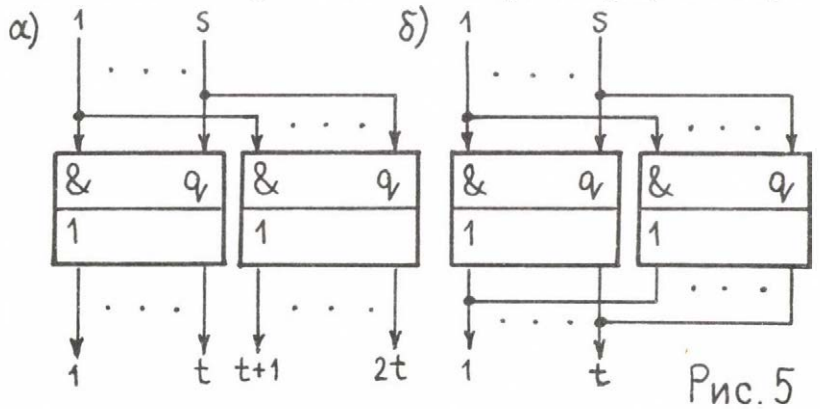


Рис. 5

если в строке h таблицы переходов МПА записан переход из a_m в a_s под действием входного сигнала x_h, то этому переходу соответствует терм ПЛМ (s,t,q), реализующий конъюнкцию $F_h = A_m \cdot X_h = (\bigwedge_{r=1}^{R} T_r^{e_{mr}})(\bigwedge_{t=1}^{T} x_{ht}^{e_{ht}})$. Здесь A_m - конъюнкция, соответствующая коду $K(a_m)$ состояния a_m; $e_{mr}, e_{ht} \in \{0,1\}$; $x_{ht}^0 = \bar{x}_{ht}$, $x_{ht}^1 = x_{ht}$; $T_r^0 = \bar{T}_r$, $T_r^1 = T_r$. Выходная переменная y_n определяется выражением: $y_n = \bigvee_{h=1}^{H} c_{nh} \cdot F_h$, где $c_{nh} = 1$, если $y_n = 1$ на h-м переходе, и $c_{nh} = 0$ в противном случае. Например, в табл. 1 $y_2 = 1$ на первом и седьмом переходах МПА, поэтому $y_2 = F_1 \vee F_7$. Подобным образом определяется и функция возбуждения r-го элемента памяти D_r, т.е. $D_r = \bigvee_{h=1}^{H} c_{rh} \cdot F_h$, где $c_{rh} = 1$, если $D_r = 1$ на r-м переходе, и $c_{rh} = 0$ в противном случае.

Если справедливо хотя бы одно из выражений: $L + R > s$, $N + R > t$, $H > q$, то для реализации МПА потребуется несколько соединенных между собой ПЛМ (s,t,q). Соединение ПЛМ (s,t,q) с целью получения схемы с числом входов, выходов или термов, превышающими параметры s,t и q, принято называть расширением ПЛМ (s,t,q) по входам, выходам или термам соответственно [5]. Для расширения ПЛМ (s,t,q) по выходам в k раз достаточно соединить одноименные входы k ПЛМ (s,t,q). Пример расширения ПЛМ (s,t,q) по выходам в два раза приведен на рис.5,а, полученная в результате схема эквивалентна ПЛМ $(s,2t,q)$. При

расширении ПЛМ (s,t,q) по термам в k раз необходимо соединить как одноименные входы, так и одноименные выходы всех k ПЛМ (s,t,q) [5]. На рис. 5,б показан пример расширения ПЛМ (s,t,q) по термам в два раза, полученная в результате схема эквивалентна ПЛМ $(s,t,2q)$. Расширение ПЛМ (s,t,q) по входам не так тривиально и определяется спецификой функций, которые должны быть реализованы [5,6].

2.2. Реализация МПА с $L+R \le s$, $N+R > t$ и $H > q$

Такая реализация МПА требует расширения ПЛМ (s,t,q) по выходам и термам, для чего необходимо $u_1 = \delta c$ ПЛМ (s,t,q), где $\delta =]\frac{N+R}{t}[$ и $c =]\frac{H}{q}[$ - коэффициенты расширения по выходам и термам соответственно.

Минимизация числа ПЛМ в этом случае возможна, если таблицу переходов МПА удастся разбить на части, каждая из которых реализуется тривиальным образом на одной ПЛМ (s,t,q) . С этой целью введем на множестве переходов (строк таблицы) МПА $W = \{w_1, \dots, w_H\}$ отношение τ такое, что $w_i \tau w_j \Leftrightarrow Y(w_i) \cap Y(w_j) \neq \emptyset$, где $Y(w_i)$ и $Y(w_j)$ - выходные сигналы (микрокоманды) на i -м и j -м переходах $(i \neq j)$. Граф Γ_τ отношения τ для рассматриваемого примера (табл. 1) приведен на рис. 6. Отношение τ определяет разбиение π_τ , в каждый блок которого попадают вершины из одной компоненты связности Γ_τ . В примере $\pi_\tau = \overline{1.3.7}, \overline{2}, \overline{4.5.12.13}, \overline{6.9}, \overline{8}, \overline{10}, \overline{11}$. Из определения отношения τ ясно, что каждому блоку B^i_τ разбиения π_τ может быть поставлен в соответствие блок $B^i_{\tau_y}$ разбиения π_{τ_y} на множестве выходных переменных $Y = \{y_1, \dots, y_N\}$, т.е. отношение τ определяет разбиение π_{τ_y} . В примере

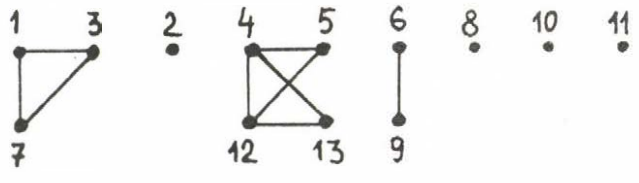


Рис. 6

$\pi_{\tau_y} = \overline{y_1 y_2 y_4}, \overline{y_8 y_9}, \overline{y_5}, \overline{y_7 y_{10}}, \overline{y_3 y_6}$. Обозначим через $q^i_{\tau} = |B^i_{\tau}|$ и $t^i_{\tau} = |B^i_{\tau_y}|$ – числа элементов в блоках B^i_{τ} и $B^i_{\tau_y}$ соответственно. Тогда, если для любых i справедливо $(q^i_{\tau} \leq q) \,\&\, (t^i_{\tau} \leq t - R) = 1$, то каждый блок разбиения π_{τ} может быть реализован на одной ПЛМ (s,t,q) , и для реализации МПА потребуется K ПЛМ (s,t,q) , где K – число блоков в разбиении π_{τ} (рис. 7). Если для некоторых пар блоков i и j $(i \neq j)$: $(q^i_{\tau} + q^j_{\tau} \leq q) \,\&\, (t^i_{\tau} + t^j_{\tau} \leq t - R) = 1$, то возникает задача нахождения разбиения $\pi'_{\tau} > \pi_{\tau}$ путем объединения блоков разбиения π_{τ} .

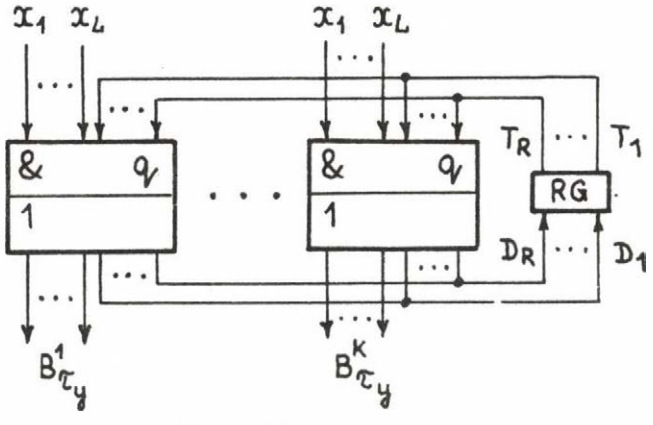


Рис. 7

Число ПЛМ, необходимых для реализации МПА на основе расширения ПЛМ (s,t,q) можно существенно сократить с помощью кодирования микрокоманд (МК). Закодируем каждую МК Y_t $(t = 0,1,\ldots,T)$ двоичным кодом $K(Y_t) = (e_{t1}, \ldots, e_{tD})$, $e_{td} \in \{0,1\}$, $D =]\log_2(T+1)[$. Обозначим через $B_t = \bigwedge_{d=1}^{D} q_d^{e_{td}}$ конъюнкцию, соответствующую коду $K(Y_t)$, $q_d^0 = \bar{q}_d$, $q_d^1 = q_d$. Функция q_d определяется выражением $q_d = \bigvee_{h=1}^{H} c_{dh} F_h$, где $c_{dh} = 1$, если d -й разряд кода МК на переходе h равен единице, и $c_{dh} = 0$ в противном случае; как и ранее, F_h – терм на h -м переходе. Тогда выходная переменная y_n будет равна: $y_n = \bigvee_{t=1}^{T} c_{nt} \cdot B_t$, где $c_{nt} = 1$, если $y_n \in Y_t$, и $c_{nt} = 0$ в противном случае. Выражения для y_n $(n = 1,\ldots,N)$ получаются непосредственно из списка МК. Например, так как в табл. 1 y_4 входит только в Y_1 , Y_4 и Y_5 , то $y_4 = B_1 \vee$

$\vee B_4 \vee B_5$. Таким образом, при кодировании МК МПА реализуется на ПЛМ (s,t,q) в виде структуры, изображенной на рис. 8.

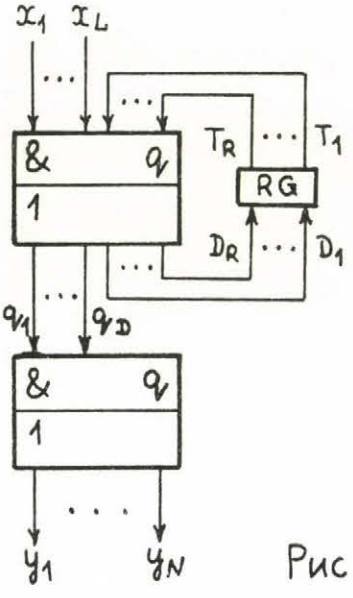


Рис. 8

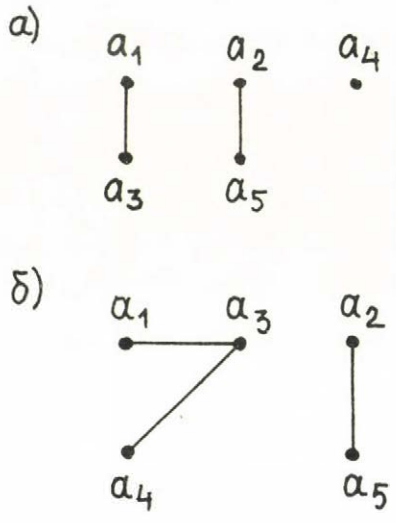


Рис. 9

При этом потребуется $u_2 = B_1c_1 + a_2B_2c_2$ ПЛМ (s,t,q) , где $B_1 =]\frac{D+R}{t}[$, $c_1 = c =]\frac{H}{q}[$, $a_2 =]\frac{D}{s}[$, $B_2 =]\frac{N}{t}[$ и $c_2 =]\frac{E}{q}[$. Как правило, $D \le s$, поэтому $a_2 = 1$. Для сокращения числа термов E необходимо так закодировать МК Y_t $(t = 0,1,\dots,T)$, чтобы минимизировалось число различных конъюнкций в выражениях для y_n $(n = 1,\dots,N)$. Тогда при расширении по выходам с числом ПЛМ, равным B_2 , необходимо разбить множество $Y = \{y_1,\dots,y_N\}$ на B_2 блоков, каждый из которых требует для реализации менее q термов, что приведет к $c_2 = 1$. Таким образом, кодирование МК дает выигрыш при $Bc > B_1c + B_2$. Поскольку обычно $B_1 = 1$ а $B_2 \approx B$, кодирование МК целесообразно при $Bc > B + c$.

2.3. Реализация МПА с $L+R>s$

Покажем, что синтез МПА с $L+R>s$ может быть сведен к рассмотренным выше способам синтеза.

Пусть $X(a_m) = \{x_{m1},\dots,x_{mG_m}\}$ - множество входных переменных в массиве переходов из состояния a_m . Введем на множестве $A = \{a_1,\dots,a_M\}$ состояний МПА S отношение ω такое, что $a_i \,\omega\, a_j \Leftrightarrow X(a_i) \cap X(a_j) \neq \emptyset$ $(i \neq j)$. Граф Γ_ω для нашего

примера показан на рис. 9,а. Граф Γ_ω определяет разбиение π_ω на множестве A такое, что в один блок π_ω попадут все состояния из одной компоненты связности Γ_ω, а число блоков равно числу компонент Γ_ω. В примере $\pi_\omega = \overline{a_1 a_3}, \overline{a_2 a_5}, \overline{a_4}$. Из определения ω ясно, что каждому блоку B^i_ω разбиения π_ω можно поставить в соответствие блок $B^i_{\omega_x}$ разбиения π_{ω_x} на множестве $X = \{x_1, \ldots, x_4\}$ входных переменных МПА. В примере $\pi_{\omega_x} = \overline{x_2 x_3 x_6}, \overline{x_1 x_4 x_5}$. Таким образом, отношение ω определяет разбиение таблицы переходов МПА. Пусть $G_x = \max_i |B^i_{\omega_x}|$, где $|B^i_{\omega_x}|$ – число элементов в блоке $B^i_{\omega_x}$. Тогда, при $G_x \le s - R$ каждый подмассив переходов МПА, определяемый разбиением π_ω, может быть реализован на ПЛМ (s,t,q) с использованием расширения по выходам или термам. При такой реализации было бы неплохо, если разделятся и выходные переменные, что выполнится, если $Y(B^p_\omega) \cap Y(B^q_\omega) = \varnothing$, где $Y(B^p_\omega)$ и $Y(B^q_\omega)$ – множества выходных переменных в подмассивах переходов из состояний, принадлежащих блокам B^p_ω и B^q_ω соответственно, $p \neq q$. С этой целью на множестве A введем отношение α такое, что $a_i \alpha a_j \Leftrightarrow (X(a_i) \cap X(a_j) \neq \varnothing) \vee (Y(a_i) \cap Y(a_j) \neq \varnothing) = 1$, где $Y(a_i)$ и $Y(a_j)$ – множества выходных переменных в массивах переходов из состояний a_i и a_j соответственно. Граф Γ_α отношения α для нашего примера приведен на рис. 9,б. Отношение α определяет соответствующие ему разбиения π_α, π_{α_x} и π_{α_y}. Для нашего примера $\pi_\alpha = \overline{a_1 a_3 a_4}, \overline{a_2 a_5}$; $\pi_{\alpha_x} = \overline{x_2 x_3 x_6}, \overline{x_1 x_4 x_5}$; $\pi_{\alpha_y} = \overline{y_1 y_2 y_4 y_5 y_7 y_{10}}, \overline{y_3 y_6 y_8 y_9}$. Если в табл. 1 переставить строки, скомпоновав вместе те из них, которые соответствуют одному блоку разбиения π_α, то можно получить табл. 2. Обозначим через s^i_α, t^i_α и q^i_α числа элементов в блоках $B^i_{\alpha_x}$, $B^i_{\alpha_y}$ и B^i_α соответственно. Тогда, как и выше, если для некоторых пар блоков i и j $(i \neq j)$: $(s^i_\alpha + s^j_\alpha \le s - R) \,\&\, (t^i_\alpha + t^j_\alpha \le t - R) \,\&\, (q^i_\alpha + q^j_\alpha \le q) = 1$, то возникает задача нахождения разбиения $\pi'_\alpha > \pi_\alpha$ путем объединения блоков разбиения π_α при выполнении условий $s^i_{\alpha'} \le s - R$, $t^i_{\alpha'} \le t - R$ и $q^i_{\alpha'} \le q$. Построенная по таблице 2 реализация МПА на ПЛМ $(6,6,10)$ приведена на рис. 10, где выходные переменные, встречающиеся в табл. 2 всегда вместе

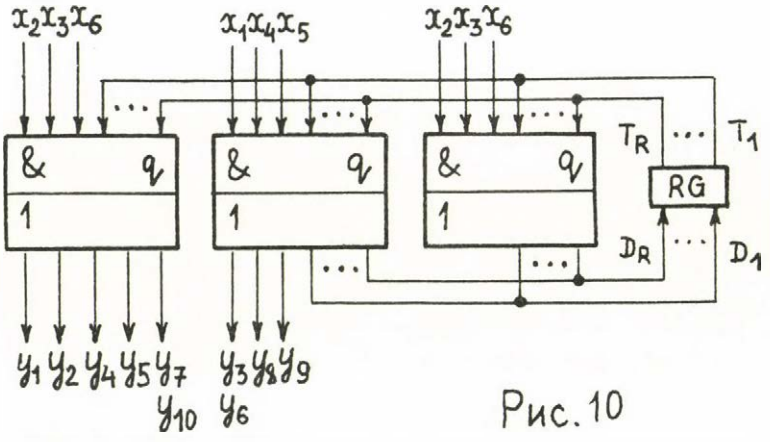


Рис. 10

($\{y_3, y_6\}$ и $\{y_7, y_{10}\}$), снимаются с одного выхода ПЛМ. Как видно из рис. 10, реализация первого блока разбиения π_1 (первого подмассива табл. 2) потребовала расширения по выходам.

В работе [1] рассмотрен способ замены множества входных переменных $X = \{x_1, \ldots, x_L\}$ МПА S множеством переменных $P = \{p_1, \ldots, p_G\}$. Пусть $X(a_m) = \{x_{m1}, \ldots, x_{mG_m}\}$, а $G = \max\limits_m G_m$. Ясно, что $G \leq L$, однако на практике, как правило, $G << L$. Тогда синтез МПА с $L+R>S$ можно свести к синтезу МПА с $G+R \leq S$ (рис. 11). Для этого образуем новое

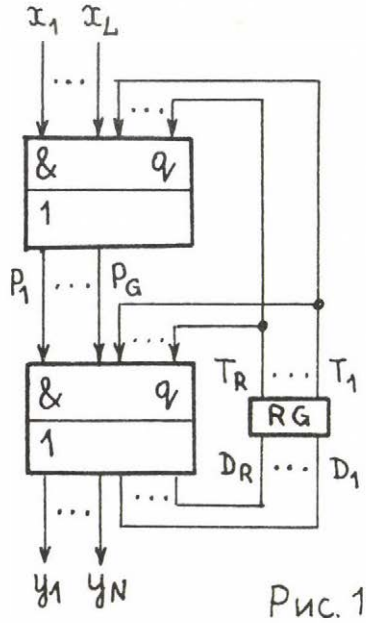


Рис. 11

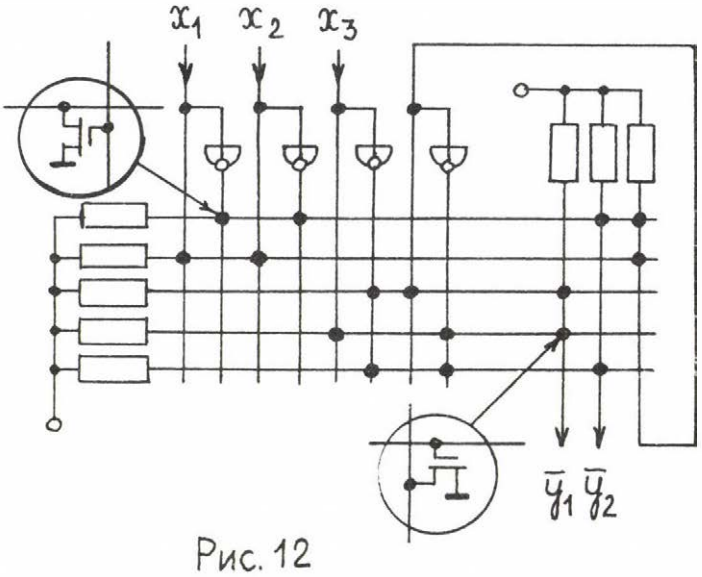


Рис. 12

множество переменных $P = \{p_1, \ldots, p_G\}$ и для каждого a_m ($m = 1, \ldots, M$) построим инъективную функцию $f_m : X(a_m) \to P$, которая определяется следующим образом: в состоянии a_m переменная $x_\ell \in X(a_m)$ заменяется переменной p_g такой, что $x_\ell = p_g$ при $A_m = 1$, где A_m, как и ранее, конъюнкция, соответствующая коду $K(a_m)$. Таким образом, переменные p_g ($g = 1, \ldots, G$) определяются выражением: $p_g = \bigvee_{m=1}^{M} (\bigvee_{\ell=1}^{L} c_{m\ell} x_\ell) A_m$, где $c_{m\ell} = 1$, если в состоянии a_m $f(x_\ell) = p_g$, и $c_{m\ell} = 0$ в противном случае. Один из возможных вариантов замены переменных для нашего примера приведен в табл. 3. В этой таблице на пересечении строки a_m и столбца p_g записан элемент x_ℓ, если $p_g = f(x_\ell)$. Выражения для p_g могут быть получены непосредственно из таблицы, подобной табл. 3. Так, например, из табл. 3 :

Таблица 3

a_m \ p_g	p_1	p_2	p_3
a_1	x_2	x_3	–
a_2	–	–	x_1
a_3	x_6	x_3	–
a_4	–	–	–
a_5	x_4	x_5	x_1

Таблица 4

T_3 \ T_1T_2	00	01	11	10
0	a_1	a_2	a_5	a_3
1		a_4		

$p_2 = A_1 x_3 \vee A_3 x_3 \vee A_5 x_5 \vee [A_2 x_3 \vee A_4 x_3 \vee A_2 x_5 \vee A_4 x_5]$. Здесь выражения, записанные в квадратных скобках, соответствуют прочеркам в табл.3, где функция p_2 не определена, и могут быть использованы при минимизации p_2. Если состояния в примере закодированы так, как показано в табл. 4, то с учетом неиспользуемых кодовых комбинаций получим: $p_1 = \overline{T}_1 x_2 \vee T_1 \overline{T}_2 x_6 \vee T_2 x_4$; $p_2 = \overline{T}_2 x_3 \vee T_2 x_5$; $p_3 = x_1$. Ясно, что в дизъюнктивной нормальной форме системы функций p_g ($g = 1, \ldots, G$) каждый терм содержит только одну входную переменную. Эксперименты над МПА показали, что число различных термов V в выражениях p_g ($g = 1, \ldots, G$) не превышает $1{,}2L$. В нашем примере $V = L = 6$. Так как G всегда меньше t, при реализации

P_g на ПЛМ (s,t,q) возникает задача нахождения разбиения π_x множества входных переменных на $K \leq]\frac{L}{S-R}[$ блоков при условии, что входные переменные κ -го блока $(\kappa = 1,\dots,K)$ входят не более чем в q термов выражений P_g $(g = 1,\dots,G)$. Таким образом, для реализации P_g потребуется $K \leq]\frac{L}{S-R}[$ ПЛМ (s,t,q).

2.4. Использование узлов в граф-схеме алгоритма

Нетрудно показать, что на ПЛМ (s,t,q) может быть реализована не только ДНФ системы булевых функций, но и скобочная форма, содержащая скобки произвольной глубины. Так, например, на рис. 12 иллюстрируется реализация на ПЛМ системы функций:

$$y_1 = \overline{x_1x_2 \vee \bar{x}_1\bar{x}_2\,\bar{x}_3} \vee (x_1x_2 \vee \bar{x}_1\bar{x}_2)\,x_3\,;$$
$$y_2 = x_1x_2 \vee \overline{x_1x_2 \vee \bar{x}_1\bar{x}_2}\,x_3\,.$$

В работе [4] используется понятие узла в граф-схеме алгоритма для многоуровневой декомпозиции логической схемы МПА. Так как узлы уменьшают длину таблицы переходов (на практике это почти всегда имеет место), то соответственно и уменьшается число термов при реализации МПА на ПЛМ (s,t,q). Кроме того, узлы сокращают длину конъюнкций в столбце X_h таблицы переходов, а, следовательно, и число элементов в множестве $X(a_m)$ $(m = 1,\dots,M)$. Это, в свою очередь, увеличивает число блоков в разбиениях π_ω и π_α, что зачастую позволяет избежать расширения ПЛМ (s,t,q) по входам и выходам при реализации МПА. На рис. 13 условно изображен фрагмент граф-схемы алгоритма, содержащей узел Q, где $X(a_m,Q)$, $X(a_n,Q)$ и $X(Q,a_s)$ - конъюнкции, соответствующие путям из состояний a_m, a_n в узел Q и из узла Q в состояние a_s. Тогда микрокоманда Y_g определяется выражением: $Y_g = \Phi(Q)\cdot X(Q,a_s)$, где $\Phi(Q) = A_m\cdot X(a_m,Q) \vee A_n X(a_n,Q)$ - функция узла Q. Ясно, что для построения Y_g необходимо функцию $\Phi(Q)$ завести с выхода ПЛМ на её вход. В общем виде реализация МПА с узлами на ПЛМ (s,t,q) будет иметь вид, изображенный на рис. 14.

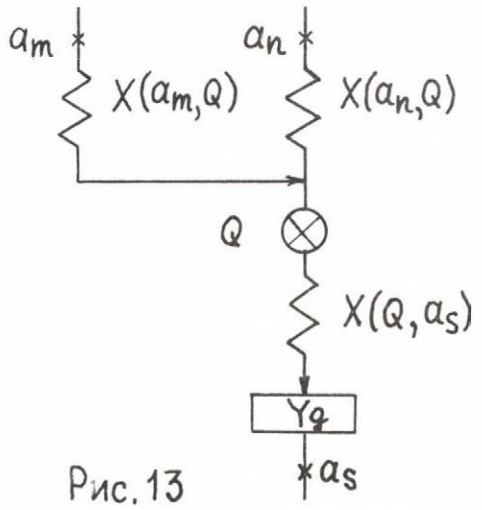


Рис. 13

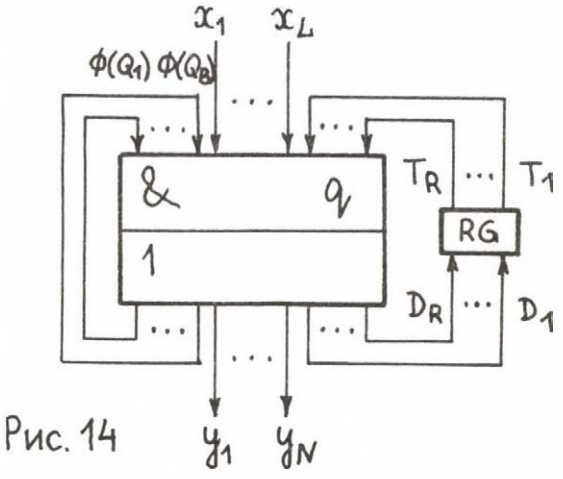


Рис. 14

3. СИНТЕЗ МИКРОПРОГРАММНЫХ АВТОМАТОВ НА ПЛМ (s,t,q,r)

Реализация МПА на ПЛМ (s,t,q) требует дополнительно схем памяти, кроме того, поскольку для связи с элементами памяти МПА используются R входов и R выходов ПЛМ (s,t,q), часто необходимы дополнительные ПЛМ для организации расширения по входам и выходам. Использование ПЛМ с памятью (рис. 3) в сочетании с методами декомпозиции из работы [7] позволяет сократить общее число ПЛМ, необходимых для реализации заданного МПА.

Ясно, что если в МПА $L \le s$, $N \le t$ и $H \le q$, то он может быть реализован тривиальным образом на одной ПЛМ (s,t,q,r). В случае, когда в МПА только $N > t$, а $L \le s$ и $H \le q$, для его реализации требуется $u =]\frac{N}{t}[$ ПЛМ (s,t,q,r), так как необходимо расширение ПЛМ (s,t,q,r) по выходам в u раз, что сводится к соединению соответствующих входов всех u ПЛМ (s,t,q,r). Таким образом, у всех u ПЛМ (s,t,q,r) матрицы M_1 реализуют одни и те же термы (число их равно $H \le q$), а на ut выходах матриц M_2 этих ПЛМ (s,t,q,r) формируется множество $Y = \{y_1,\dots,y_N\}$ выходных переменных. Тогда это множество Y должно быть разбито на $u =]\frac{N}{t}[$ блоков, и выходные переменные, принадлежащие данному блоку разбиения, формируются в одной из u ПЛМ (s,t,q,r).

Реализация МПА с $L > S$ или $H > q$ на ПЛМ (s,t,q,r) основана на предложенных в [7] методах декомпозиции МПА, в результате чего для заданного автомата S по выбранному разбиению $\pi = \{A^1, \ldots, A^n\}$ на множестве A его состояний строится сеть из компонентных автоматов $S^m = (B^m, X^m, Y^m, \delta^m, \lambda^m)$, $m = 1, \ldots, n$, каждый из которых затем реализуется на ПЛМ (s,t,q,r). В качестве примера будем рассматривать декомпозицию МПА, заданного табл. 1. Пусть в нашем примере $\pi = \{A^1, A^2\}$, где $A^1 = \{a_1, a_3, a_4\}$, $A^2 = \{a_2, a_5\}$. Вопросы, связанные с выбором разбиения π, будут рассмотрены ниже.

Паре (S, π) поставим в соответствие сеть C из n компонентных МПА $S^m = (B^m, X^m, Y^m, \delta^m, \lambda^m)$, $m = 1, \ldots, n$, определяемых следующим образом:

1. $B^m = A^m \cup \{b_m\}$, где A^m – m-ый блок разбиения π, а b_m – дополнительное состояние в автомате S^m (такое состояние вводится в каждом компонентном автомате). Таким образом, в примере сеть будет состоять из двух компонентных автоматов S^1 и S^2, множества состояний которых определяются блоками разбиения π: $B^1 = A^1 \cup \{b_1\} = \{b_1, a_1, a_3, a_4\}$; $B^2 = A^2 \cup \{b_2\} = \{b_2, a_2, a_5\}$.

2,3. Определим δ^m и λ^m. Пусть в автомате S есть переход из a_i в a_j под действием входного сигнала X_h с выдачей выходного сигнала Y_t: $\delta(a_i, X_h) = a_j$, $\lambda(a_i, X_h) = Y_t$ и пусть $a_i \in A^m$, т.е. a_i – состояние компонентного автомата S^m. Возможны два случая.

а) $a_j \in A^m$, т.е. a_j – тоже состояние компонентного автомата S^m. Тогда положим $\delta^m(a_i, X_h) = \delta(a_i, X_h) = a_j$; $\lambda^m(a_i, X_h) = \lambda(a_i, X_h) = Y_t$;

б) $a_j \in A^p$, т.е. a_j – состояние компонентного автомата S^p $(p \neq m)$. Таким образом, a_i и a_j находятся в разных компонентных автоматах. Будем говорить, что в этом случае компонентный автомат S^m возбуждает компонентный автомат S^p. Обо-

значим через $Z = \{z_1, \ldots, z_T\}$ множество связей между компонентными автоматами в сети C, по которым они возбуждают друг друга. Тогда функции переходов и выходов в S^m и S^p определяются следующим образом: $\delta^m(a_i, x_h) = b_m$, $\lambda^m(a_i, x_h) = Y_t \cup \{z_{ih}\}$; $\delta^p(b_p, z_{ih}) = \delta(a_i, x_h) = a_j$, $\lambda^p(b_p, z_{ih}) = Y_0$, где Y_0 - векторный выходной сигнал, все компоненты которого равны нулю. Здесь $z_{ih} \in Z$ - сигнал связи компонентных автоматов S^m и S^p.

4. $X^m = \tilde{X}^m \cup Z_x^m$, где $\tilde{X}^m = \bigcup_{a_i \in A^m} X(a_i)$ - множество входных переменных, встречающихся в массиве переходов из состояний множества A^m, а Z_x^m - множество входных сигналов связи в S^m при его возбуждении другими компонентными автоматами. В нашем примере $\tilde{X}^1 = \{x_2, x_3, x_6\}$, $Z_x^1 = \{z_2, z_3\}$, $\tilde{X}^2 = \{x_1, x_4, x_5\}$ и $Z_x^2 = \{z_1\}$.

5. $Y^m = \tilde{Y}^m \cup Z_y^m$, где $\tilde{Y}^m = \bigcup_{a_i \in A^m} Y(a_i)$ - множество выходных переменных, встречающихся в массиве переходов из состояний $a_i \in A^m$, а Z_y^m - множество сигналов связи из S^m, которыми он возбуждает другие компонентные автоматы. В нашем примере $\tilde{Y}^1 = \{y_1, y_2, y_4, y_5, y_7, y_{10}\}$, $Z_y^1 = \{z_1\}$, $\tilde{Y}^2 = \{y_3, y_6, y_8, y_9\}$ и $Z_y^2 = \{z_2, z_3\}$.

Структурные таблицы компонентных автоматов S^1 и S^2 приведены в таблицах 5 и 6 соответственно. В отличие от таблицы переходов структурная таблица МПА содержит три дополнительных столбца: $K(a_m) = (e_{m1}, \ldots, e_{mR})$ - код состояния a_m $K(a_s) = (e_{s1}, \ldots, e_{sR})$ - код состояния a_s, $e_{mr}, e_{sr} \in \{0,1\}$, $\tilde{F}_h$ - множество функций возбуждения элементов памяти на h-м переходе. Поясним построение функций переходов и выходов этих автоматов. В автомате S есть переход (a_1, a_3) под действием входного сигнала $x_2 x_3$ с выдачей выходного сигнала $Y_4 = \{y_1, y_2, y_4\}$ (первая строка табл. 1). Так как $a_1 \in A^1$ и $a_3 \in A^1$, то в автомате S^1 также будет переход из a_1 в a_3 под действием того же входного сигнала с выдачей того же выходного сигнала: $\delta^1(a_1, x_2 x_3) = a_3$, $\lambda^1(a_1, x_2 x_3) =$

Таблица 5

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Y_t	$\tilde{F}_h$	h
a_1	01	a_3	00	$x_2 x_3$	$y_1 y_2 y_4$	–	1
		a_1	01	$x_2 \bar{x}_3$	–	D_2	2
		b_1	11	$\bar{x}_2$	$y_4 z_1$	$D_1 D_2$	3
a_3	00	a_4	10	x_3	y_5	D_1	4
		a_3	00	$\bar{x}_3 \bar{x}_6$	$y_2 y_4$	–	5
		a_4	10	$\bar{x}_3 x_6$	$y_7 y_{10}$	D_1	6
a_4	10	a_1	01	1	y_5	D_2	7
b_1	11	a_3	00	z_2	–	–	8
		a_1	01	z_3	–	D_2	9

Таблица 6

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Y_t	$\tilde{F}_h$	h
a_2	00	a_5	01	x_1	$y_8 y_9$	D_2	1
		b_2	10	$\bar{x}_1$	$y_9 z_2$	D_1	2
a_5	01	a_5	01	x_1	$y_3 y_6$	D_2	3
		b_2	10	$\bar{x}_1 x_4 x_5$	z_3	D_1	4
		a_2	00	$\bar{x}_1 \bar{x}_4 x_5$	$y_8 y_9$	–	5
		a_5	01	$\bar{x}_1 x_5$	y_8	D_2	6
b_2	10	a_2	00	z_1	–	–	7

$= \{y_1, y_2, y_4\}$ (первая строка в табл. 5). В автомате S есть переход (a_2, a_3) под действием входного сигнала $\bar{x}_1$ с выдачей выходного сигнала $Y_9 = \{y_9\}$. Так как $a_2 \in A^2$, а $a_3 \in A^1$, то в автомате S^2 : $\delta^2(a_2, \bar{x}_1) = b_2$, $\lambda^2(a_2, \bar{x}_1) = \{y_9, z_2\}$ (вторая строка в табл. 6). При этом в автомате S^1 (табл. 5) будет переход (b_1, a_3) под действием сигнала связи z_2 с выдачей выходного сигнала Y_0 : $\delta^1(b_1, z_2) = a_3$, $\lambda^1(b_1, z_2) = Y_0$. Интересно заметить, что если автомат S^m возбужден (находится в состоянии типа $a_i \in A^m$), то все другие компонентные автоматы находятся в состоянии типа b_ρ $(\rho = 1, \ldots, m-1, m+1, \ldots, n)$. Ясно, что каждый раз не может возбуждаться более одного автомата, т.е. компонентные автоматы работают во времени последовательно за исключением моментов возбуждения.

Очевидно, что если для любого m :

$$(|X^m| \leqslant s) \,\&\, (|Y^m| \leqslant t) \,\&\, (H^m \leqslant q) = 1 \qquad (I)$$

то сеть C может быть реализована тривиальным образом на n ПЛМ (s, t, q, r), каждая из которых реализует компонентный

автомат S^m. В выражении (1) H^m - число переходов (строк в структурной таблице) автомата S^m. Из выражения (1) ясно, что в общем случае для уменьшения числа ПЛМ (s,t,q,r) необходимых для реализации сети C, важно, чтобы в результате декомпозиции автомата S у любых двух компонентных автоматов S^m и $S^p (m \neq p)$ множества входных (X^m и X^p) и выходных (Y^m и Y^p) переменных не пересекались, или это пересечение было бы минимальным. Поэтому для определения разбиения π при декомпозиции МПА можно воспользоваться отношениями ω (для разделения входных переменных у компонентных автоматов) или α (для разделения и входных и выходных переменных), определенными выше. В примере при декомпозиции МПА, заданного табл. 1, было использовано разбиение π_α. Если в сети C найдутся m и p такие, что

$$(|X^m \cup X^p| \leq s) \,\&\, (|Y^m \cup Y^p| \leq t) \,\&\, (H^m + H^p \leq q) = 1, \quad (2)$$

то компонентные автоматы S^m и S^p могут быть реализованы на одной ПЛМ (s,t,q,r). Следовательно, при выполнении выражения (2) возникает задача нахождения объединения компонентных автоматов в $n' < n$ блоков, при котором число ПЛМ (s,t,q,r), необходимых для реализации сети C', было бы минимальным. Данная задача означает нахождение разбиения $\pi'_\alpha > \pi_\alpha$, декомпозиция по которому даст сеть C' из компонентных автоматов $S^{m'} (m' = 1,\dots,n')$ с минимальным n'.

Нетрудно видеть, что при реализации компонентных автоматов S^1 и S^2 на ПЛМ $(6,6,10,3)$ условие (1) выполняется, а условие (2) - нет. Поэтому для их реализации потребуется две ПЛМ $(6,6,10,3)$ (рис.15). В схеме на рис. 15 выходные переменные, встречающиеся в структурных таблицах всегда вместе, снимаются с одного выхода ПЛМ $(6,6,10,3)$. В примере это $\{y_3, y_6\}$ и $\{y_7, y_{10}\}$. На рис. 16 показана схема ПЛМ $(6,6,10,3)$, реализующая компонентный автомат S^1. Отметим, что реализация МПА, заданного табл. 1, на ПЛМ $(6,6,10)$ требует не менее трех ПЛМ плюс дополнительные схемы памяти.

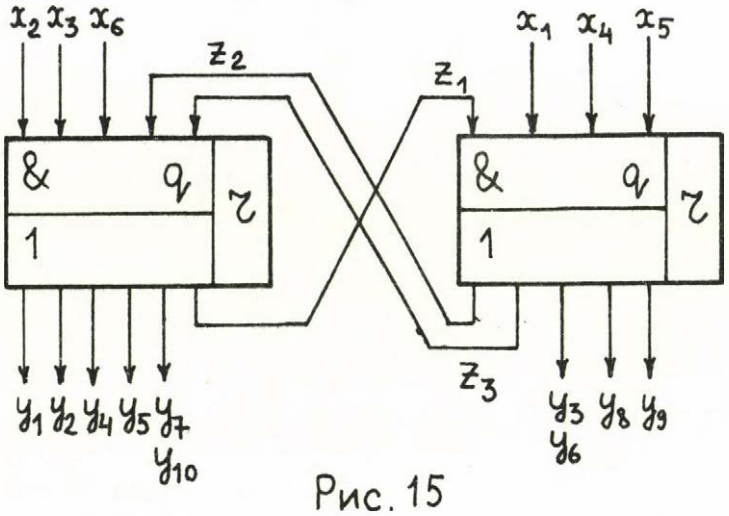

x2 x3 x6
z2
z1
x1 x4 x5
&
q
z
1
&
q
z
1
z3
y1 y2 y4 y5 y7
y10
y3 y8 y9
y6

Рис. 15

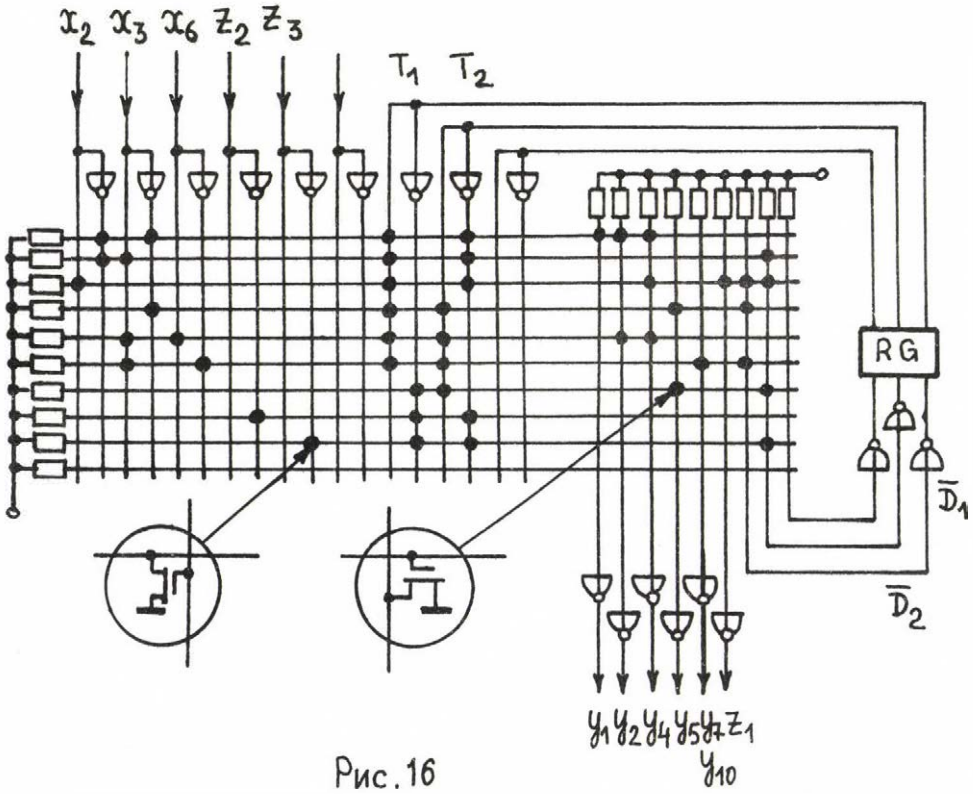

x2 x3 x6 z2 z3
T1 T2
RG
D1
D2
y1 y2 y4 y5 y7 z1
y10

Рис. 16

ЛИТЕРАТУРА

1. Baranov S.I., Matrix realization of control automata. - In: Proceedings of the 2d International Symposium on Discrete Systems, Dresden, IFAC 1977, V.2.

2. Timm V. Im Brickpunkt: ROM, PROM und PLAS. Elektronik, 1976, H.5.

3. Баранов С.И., Синев В.Н. Программируемые логические матрицы в цифровых системах: Обзор. - Зарубежные радиоэлектроника, 1979, № 1.

4. Баранов С.И. Синтез микропрограммных автоматов. - Л.: Энергия, 1974.

5. Hemell A., The PLA: a' different kind' of ROM. - Electronic Design, 1976, V.24. N1.

6. Новиков С.В. Синтез схем на программируемых логических матрицах. - Автоматика и вычислительная техника, 1977, № 5.

7. Баранов С.И., Килленберг Х. Декомпозиция микропрограммных автоматов. - Автоматика и вычислительная техника, 1978, № 6.

СПИСОК АВТОРОВ

ДО ЭТОГО ВЫШЕДШИЕ СБОРНИКИ СЕРИИ
MTA SZTAKI TANULMÁNYOK

№ 21 Доклады семинара, проводимого 19-23-ого февраля 1973 г. в Венгрии во время совещания по теме 1-15.1. "Структура и методологические рекомендации по созданию систем автоматизированного проектирования дискретных устройств". Будапешт, 1974.

№ 63 Доклады семинара, проводимого 20-21 мая 1976 г. в Риге во время совещания по теме 1-15.1. "Разработка общей теории автоматов".
Будапешт, 1977.