

РЕАЛИЗАЦИЯ АВТОМАТОВ В КРИОТРОННОЙ ВЫЧИСЛИТЕЛЬНОЙ СРЕДЕ

О.А. Бандман
(Новосибирск)

При реализации логических функций и автоматов в вычислительной среде очень важным является получение достаточно простых алгоритмов перехода от заданных канонических уравнений и таблиц к составлению программы настройки вычислительной среды. Очевидно, такой переход должен включать в себя не только составление программы (определение состояния настройки для каждого элемента среды) по уже заданной логической сети, но также и элементы синтеза логических функций с учетом особенностей как вычислительной среды в целом, так и реальных элементов, из которых она построена.

В настоящей работе делается попытка использовать методы релейно-контактного синтеза и особенности сверхпроводящих схем для составления программы настройки криотронной вычислительной среды при реализации логических функций, выраженных в базисе (И, ИЛИ, НЕ), а также автоматов, заданных каноническими таблицами и уравнениями. Методика применима только к конкретному типу плоской вычислительной среды, элементами которой могут выполнять три соединительных функции ("р", "д" и "о" согласно обозначениям, введенным в [1]) и релейно-контактную функцию. Кроме того, все образованные в среде схемы подчиняют-

ся законам сверхпроводящих цепей [2]. Работа элемента такого типа, построенного на восьми криотронах, описана в [3]

Предлагаемая методика дает в результате плотное заполнение участка среды участвующими в работе схемы элементами, но не является минимальной ни с точки зрения количества функциональных элементов, ни с точки зрения общего количества элементов среды.

I. Свойства криотронных цепей

Криотрон — элемент, по своему логическому действию аналогичный нормально разомкнутому контакту реле. При пропускании через его обмотку тока определенного уровня сверхпроводимость вентили разрушается. Поскольку все соединения в криотронных схемах сверхпроводящие ($R = 0$), появление даже очень малого сопротивления в одной из нескольких параллельных ветвей аналогично её полному размыканию и вызывает полное переключение тока в параллельные сверхпроводящие ветви.

Сверхпроводящие цепи обладают некоторыми специфическими свойствами, которые необходимо учитывать и по возможности использовать при проектировании криотронных логических устройств. Основные из них следующие.

1. Носителем информации в криотронных схемах является ток, а не напряжение; состояние схемы определяется наличием или отсутствием тока в тех или иных ветвях схемы.

2. Схемы должны быть организованы так, чтобы в установившихся режимах в них всегда существовал сверхпроводящий путь для тока.

3. В криотронных схемах происходит автоматическое запоминание информации, так как состояние схемы сохраняется после снятия сигналов до тех пор, пока существует питание и охлаждение.

4. В параллельных сверхпроводящих ветвях ток распределяется обратно пропорционально их индуктивностям.

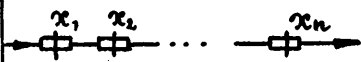
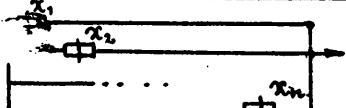
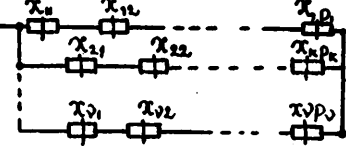
Самой существенной из перечисленных четырех особенностей для реализации автоматов является третья, использование которой существенно упрощает синтез схем с памятью.

2. Реализация логических функций, выраженных в д.н.ф.

Как показано в [4], любую функцию алгебры логики можно реализовать параллельно-последовательной схемой из двухполюсных переключающих элементов. Если учесть, что криотрон является инвертором и что полному отсутствию сопротивления в цепи соответствует "единица", а наличию сопротивления - "ноль" на выходе, то получатся приведенные в таблице I соответствия между

Т а б л и ц а I

Соединения криотронов, реализующие логические функции в д.н.ф.

Соединения криотронов	$f(x_1, x_2, \dots, x_n)$
	$\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \dots \cdot \bar{x}_n$
	$\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee \dots \vee \bar{x}_n$
	$\bigvee_{k=1}^k \bar{x}_{k1} \cdot \bar{x}_{k2} \cdot \dots \cdot \bar{x}_{kp_k}$

логическими функциями и соединениями криотронов. При реализации этих соединений в вычислительной среде каждому криотрону должен соответствовать элемент "F", а все соединения должны быть выполнены элементами "D" и "P". Относительное размещение элементов "F", "D", "P" (программа настройки [5]) легко определяется из следующего.

Пусть функция алгебры логики задана в виде

$$f(x_1, \bar{x}_1, x_2, \bar{x}_2, \dots, x_n, \bar{x}_n) = \bigvee_{k=1}^v x_{k_1} x_{k_2} \dots x_{k_{p_k}}, \quad (I)$$

где $\{x_1, \bar{x}_1, x_2, \bar{x}_2, \dots, x_n, \bar{x}_n\} = X$ - множество входных переменных и их отрицаний,

$\{x_{k_1}, x_{k_2}, \dots, x_{k_{p_k}}\} = X_k \subset X$ - подмножество переменных, входящих в k -й дизъюнктивный член (I).

Выражение (I) может быть также представлено в виде таблицы истинности, представляющей собой матрицу T размерности $v \times m$. Элементами матрицы являются

$$\begin{aligned} t_{ki} &= 1, & \text{если } x_i \in X_k, \\ t_{ki} &= 0, & \text{если } x_i \notin X_k. \end{aligned} \quad (2)$$

Если в T все единицы заменить на элементы "P", а все нули - на элементы "D" и подвести к каждому столбцу соответствующее \bar{x}_i , то получим набор строк, в каждой из которой реализуется одна из конъюнкций $x_{k_1} \cdot x_{k_2} \cdot \dots \cdot x_{k_{p_k}}$. Чтобы реализовать дизъюнкции, остается только объединить в узлы начала и концы этих строк, приписав к полученной таблице слева и справа по столбцу элементов "P".

Иными словами, чтобы получить программу настройки среды для реализации функции, заданной в виде (I) следует пользоваться следующим алгоритмом.

1. Выделить участок среды размерами $v \times m$.
2. Элементам этого участка a_{ki} поставить в соответствие элементы "P", если $x_i \in X_k$, или элементы "D", если $x_i \notin X_k$.
3. Приписать к выделенному участку слева и справа по одному столбцу элементов "P".
4. Подвести к каждому столбцу отрицание соответствующей переменной x_i .

Например, для функции $f(X) = x_1 x_2 \bar{x}_3 \vee x_1 \bar{x}_2 x_3 \vee x_4 x_5$

	x_1	x_2	\bar{x}_2	x_3	\bar{x}_3	x_4	x_5
1	1	1			1		
2	1		1			1	
3				1		1	
4		1					1

а программа изображена на рис. 1:

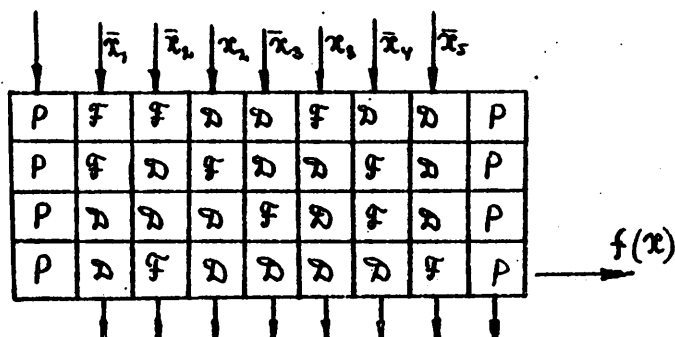


Рис. 1.

3. Реализация логических функций, выраженных в скобочной форме

При минимизации функций алгебры логики стремление уменьшить число функциональных элементов часто приводит к выражениям со скобками вида

$$f(X) = \left(\bigvee_{k=1}^n x_{k_1} x_{k_2} \dots x_{k_{p_k}} \right) \left(\bigvee_{k=1}^m x_{k_1} x_{k_2} \dots x_{k_{p_k}} \right) \dots \left(\bigvee_{k=1}^l x_{k_1} x_{k_2} \dots x_{k_{p_k}} \right). \quad (3)$$

Функция вида (3) реализуется параллельно-последовательным соединением функциональных элементов так, что каждой конъюнкции соответствует последовательное, а каждой дизъюнкции — параллельное соединение каких-либо подэлементов.

Расположение блоков друг относительно друга может быть различным: они могут располагаться и рядом друг с другом и один под другим в зависимости от того, реализуют ли они функции от различных переменных или от одних и тех же. Если две функции $f_1(x_1, x_2, \dots, x_k)$ и $f_2(x_{k+1}, \dots, x_n)$, реализуемые соответственно двумя участками среды Λ_1 и Λ_2 , объединяются знаком конъюнкции и не имеют одинаковых переменных, то блоки Λ_1 и Λ_2 лучше расположить рядом друг с другом, обеспечив их последовательное соединение разделяющим их столбцом элементов "Р". Если блоки, соответствующие двум конъюнктивным членам имеют одинаковые переменные, то более компактное расположение получается при размещении их друг под другом.

Блоки, соответствующие дизъюнктивным членам от различных переменных тоже имеет смысл располагать рядом друг с другом. Однако в этом случае, чтобы обеспечить их параллельное соединение, необходимо добавлять специальные строки, целиком состоящие из соединительных элементов.

Например, минимальная форма функции

$$f(X) = x_1 x_2 \vee x_2 x_4 \vee x_4 \bar{x}_5 \vee x_3 x_4 \vee x_1 x_3 x_5$$

имеет вид

$$f_{min} = (x_1 \vee x_4)(x_2 \vee x_3 x_5) \vee x_4 \bar{x}_5.$$

Дизъюнкции $f_1 = x_1 \vee x_4$ и $f_2 = x_2 \vee x_3 x_5$ не имеют одних и тех же переменных, поэтому соответствующие им блоки A_1 и A_2 располагаются в программе один рядом с другим. Блоки A_3 и A_4 , реализующие дизъюнктивные члены $f_3 = f_1 \cdot f_2$ и $f_4 = x_4 \bar{x}_5$ расположены один под другим. (рис. 2).

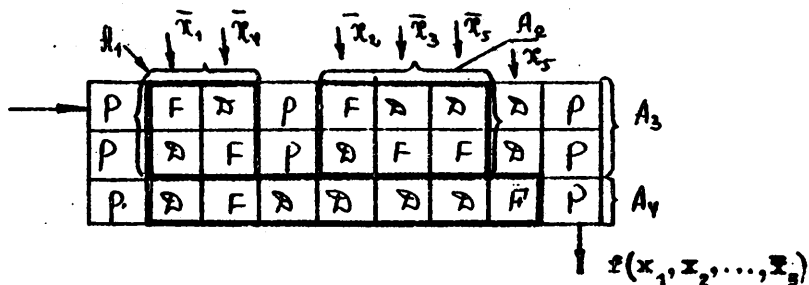


Рис. 2.

4. Каскадная реализация логических функций

Каскадная реализация в криотронной вычислительной среде соответствует такой форме записи логической функции, в которой содержатся некоторые более простые функции (подфункции) под знаком отрицания

$$f(x_1, x_2, \dots, x_n) = \varphi(x_1, x_2, \dots, x_n, \bar{x}_1, \dots, \bar{x}_k). \quad (4)$$

Здесь f_1, \dots, f_k — функции тех же переменных x_1, \dots, x_k , каждая из которых, в свою очередь, может содержать подфункции более низких порядков.

Подфункции f_1, f_2, \dots, f_k могут быть реализованы отдельно соответствующими им подсхемами (блоками), выходы которых служат входами при реализации

$$\varphi(x_1, \dots, x_k, \bar{f}_1, \dots, \bar{f}_k).$$

Составление программы при каскадной реализации следует начинать с подфункций f_1, f_2, \dots, f_k . Если они не имеют одинаковых переменных, то реализующие их блоки лучше располагать рядом, разделяя их столбцами с элементами "0". Если же подфункции имеют одинаковые переменные, их лучше расположить друг под другом, обеспечив разделение входов и выходов, либо введением разделительных строк, либо использованием различных входных и выходных столбцов. Блок, реализующий функцию следующего разряда, располагается под блоками подфункций, при этом некоторые его входы присоединяются к выходам верхних блоков.

Приведение логической функции к форме вида (4) с целью последующей каскадной реализации может быть полезным при реализации многоразрядных устройств, а иногда просто приводит к более компактной программе.

Например, если функцию

$$f(x_1, \bar{x}_1, \dots, x_6, \bar{x}_6) =$$

$$= x_1 x_4 \bar{x}_6 \vee x_2 x_4 \bar{x}_6 \vee x_3 x_4 \bar{x}_6 \vee x_1 x_5 x_6 \vee x_2 x_5 x_6 \vee x_3 x_5 \vee \bar{x}_1 \bar{x}_2$$

выразить в скобочной форме, выделив в ней подфункцию

$$f_1 = \bar{x}_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_4 \bar{x}_5,$$

$$f(x_1, \bar{x}_1, \dots, x_6, \bar{x}_6) = f_1 \bar{x}_6 \vee \bar{x}_3 \bar{x}_5 \vee \bar{x}_1 \bar{x}_2,$$

то общее число переменных сократится с 10 до 6, и функцию можно будет реализовать в два каскада. Сначала реализуется функция $f_1(\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{x}_5)$ затем непосредственно под этим блоком располагается программа для $f(\bar{f}_1, \bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_5, \bar{x}_6)$ (рис. 3)

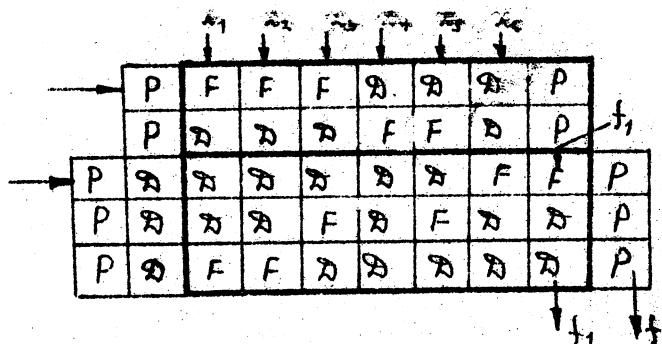


Рис. 3.

5. Применение принципа "двойной логики"

Как было уже отмечено, криотронные схемы должны быть всегда организованы так, чтобы существовал хотя бы один сверхпроводящий путь для тока. При реализации логических функций это означает, что параллельно блоку, реализующему логическую функцию, необходимо иметь еще какую-либо сверхпроводящую ветвь, в которую будет вытеснен ток питания, если $f(x_1, \dots, x_n) = 0$, а также обеспечить возможность обратного переброса тока в логическую ветвь. Это можно сделать двумя способами.

1. Организацией параллельно логическому блоку еще одной ветви для тока (так называемой "прямой" ветви) с одним криотроном, на управляющую обмотку которого подаются "рабочие" импульсы (рис. 4). Рабочие импульсы подаются после каждого нового набора переменных, и длительность их должна быть достаточной для полного переброса тока питания в логическую ветвь.

2. Применением принципа "двойной логики" (бинарного кодирования). Цепь тока питания разделяется на 2 параллельные ветви: в одной из них реализуется $f(x_1, x_2, \dots, x_n)$, в другой — $\bar{f}(x_1, \dots, x_n)$. При этом, если одна из этих ветвей образует сверхпроводящий путь, то в другой — он обязательно имеет сопротивление. После снятия входных переменных схема сохраняет свое состояние до прихода следующего воздействия (рис. 5). Иными словами, схема двойной логики образует триггер со многими входами, состояние которого

$$q(t+1) = f[x_1(t), x_2(t), \dots, x_n(t)] \quad (5)$$

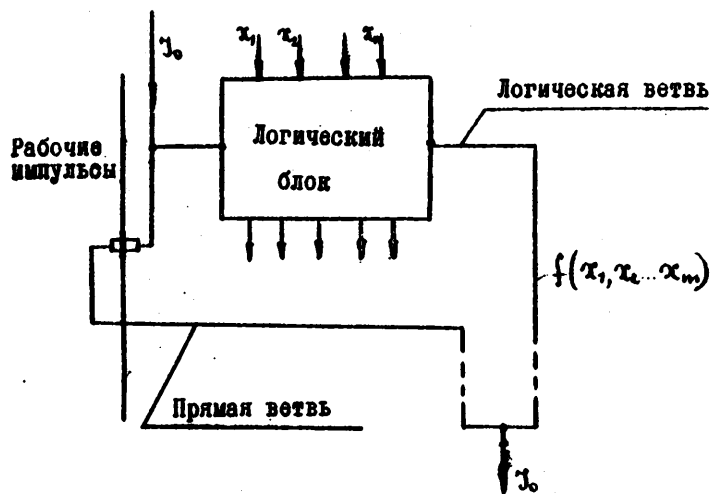


Рис. 4

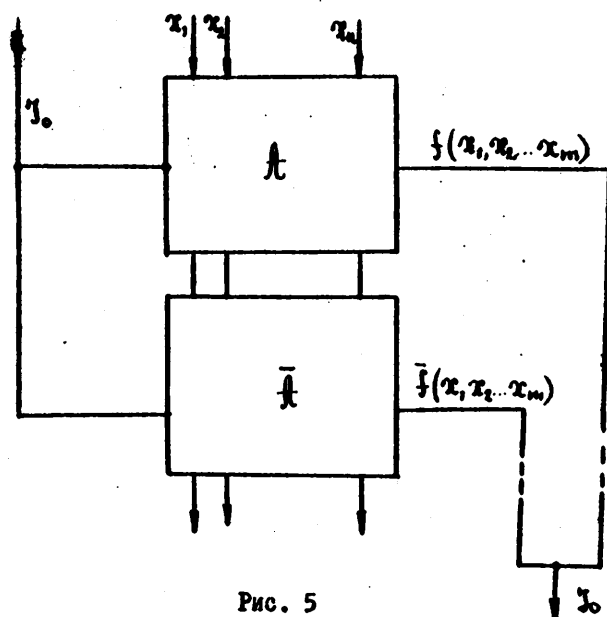


Рис. 5

Использование принципа "двойной логики" требует почти вдвое большего числа элементов среды, чем в схемах с рабочими импульсами. Поэтому при реализации комбинационных схем применение "двойной логики" нецелесообразно. Однако для реализации схем с памятью, а также сложных многовыходных многокаскадных схем, принцип "двойной логики" оказывается очень удобным, так как результат выполнения функции автоматически сохраняется до следующего такта.

Составление программы для реализации логической функции по принципу "двойной логики" сводится к построению двух программ: для $f(x_1, \dots, \bar{x}_n)$ и для $\bar{f}(x_1, \dots, \bar{x}_n)$. При этом форма выражения обеих функций и метод составления каждой программы могут быть любыми. Блоки А и \bar{A} , соответствующие обеим программам, удобно располагать друг под другом. Слева они объединяются одним "Р"-столбцом, справа каждый из них имеет свой "Р"-столбец, соответствующий своему выходу (рис. 6).

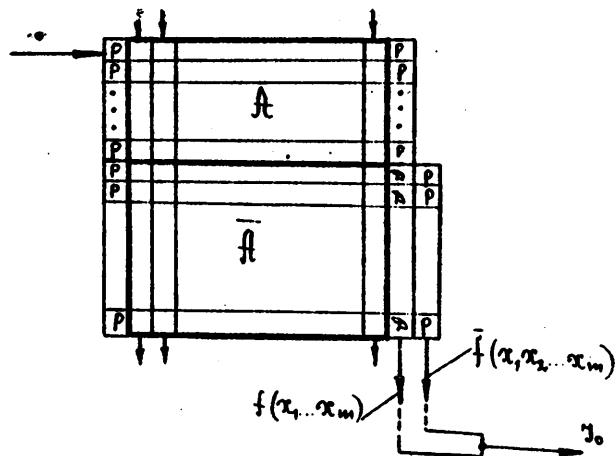


Рис. 6

Если функция выражена в с.д.н.ф.

$$\left. \begin{aligned} f(x_1, \dots, x_n) &= \bigvee_{f(\sigma_1, \dots, \sigma_n)=1} x_1^{\sigma_1} x_2^{\sigma_2} \dots x_n^{\sigma_n} \\ \bar{f}(x_1, \dots, x_n) &= \bigvee_{f(\sigma_1, \dots, \sigma_n)=0} x_1^{\sigma_1} x_2^{\sigma_2} \dots x_n^{\sigma_n} \end{aligned} \right\} \quad (6)$$

то наборы, для которых $f(x_1, \dots, x_n)=1$, образуют строки программы Λ , а наборы для которых $f(x_1, \dots, x_n)=0$, - образуют строки $\bar{\Lambda}$. Число строк программы будет равно числу наборов, а число столбцов - числу переменных и их отрицаний ($m = 2n$) плюс еще 3 "г"-столбца

$$I_1 = 2^n (2n + 3). \quad (7)$$

Если функция задана таблицей истинности, то нет необходимости записывать ее в с.д.н.ф., можно использовать непосредственно таблицу для составления программы. Для этого ее следует переписать так, чтобы на каждую переменную, а также на функции, приходилось по 2 столбца - для x_k и \bar{x}_k , для f и \bar{f} . Например, таблица истинности

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

приобретает вид

x_1	\bar{x}_1	x_2	\bar{x}_2	x_3	\bar{x}_3	f	\bar{f}
0	1	0	1	0	1	0	1
0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	0
0	1	1	0	1	0	0	1
1	0	0	1	0	1	1	0
1	0	0	1	1	0	0	1
1	0	1	0	0	1	0	1
1	0	1	0	1	0	1	0

Множество строк $S = \{s_1, s_2, \dots, s_{2^n}\}$ этой таблицы разбито на 2 непересекающихся подмножества S_1 и S_2 :

$$\left. \begin{array}{ll} s_k \in S_1 & \text{если } f(s_k) = 1 \\ s_k \in S_2 & \text{если } f(s_k) = 0 \end{array} \right\}.$$

S_1 и S_2 (без столбцов f и \bar{f}) образуют матрицы T для реализации функций f и \bar{f} соответственно. Чтобы сгруппировать строки S_1 и S_2 в две отдельные матрицы, нет необходимости переставлять их местами. Достаточно при переходе от таблицы к программе заменить в столбцах f и \bar{f} единицы - на элементы "P", а нули - на элементы "D".

Затем надо приписать справа "P"-столбец и произвести замену единиц и нулей в x -столбцах на "P" и "D" - элементы соответственно, а переменные заменить их отрицаниями. Для приведенной выше таблицы истинности программа изображена на рис.7.

Построение программы по таблице истинности по существу сводится только к расположению "P" и "D" - элементов в двух столбцах, соответствующих f и \bar{f} . Остальная программа одинакова для любой функции от n переменных, так как она отражает стандартное расположение наборов.

6. Реализация элементов памяти

В криотронных схемах существует два способа хранения информации: 1) запись незатухающего тока в сверхпроводящий контур, 2) использование свойств тока сохранять свой путь после снятия воздействия, заставившего его выбрать этот путь.

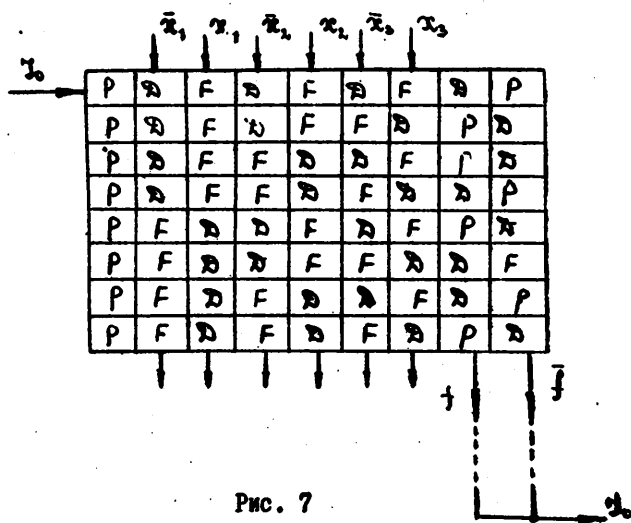


Рис. 7

И тот и другой способ используются в кривотрошной вычислительной среде. Как видно из описания схемы элемента [3], в основу организации настраиваемой памяти заложен первый способ. При реализации схем задержки в среде могут быть использованы оба способа.

При использовании записи негашущего тока необходимо реализовать в вычислительной среде схему рис. 8а, что приводит к программе из 12 элементов (рис. 8б). Чтобы за-

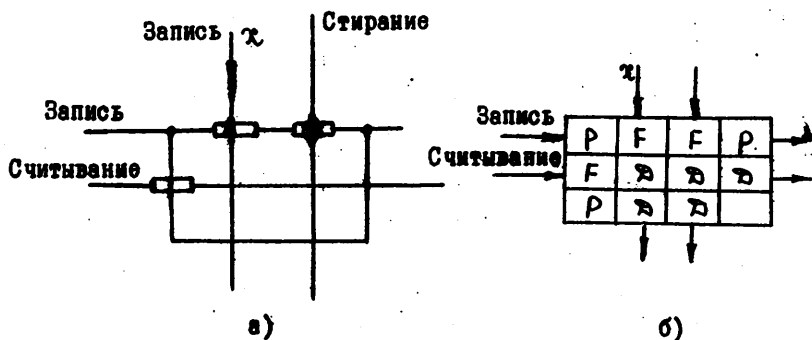
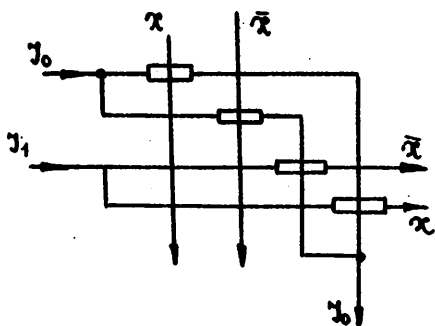


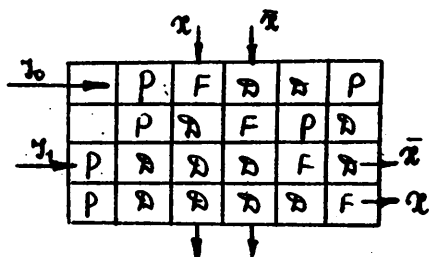
Рис. 8

писать $x = 1$, необходимо подать одновременно с ним импульс записи I_z , который должен закончиться позже, чем x . Тогда в контуре α запишется ток, который будет сохраняться до того момента, пока не придет импульс стирания. Контур α может быть как угодно длинным, и записанная информация может воздействовать на сколько угодно элементов других логических схем.

На рис. 9 (а и б) показаны схема и программа ячейки па-



а)



б)

Рис. 9

лети, использующей второй принцип. Записываемая информация x (или \bar{x}) переводит ток I_0 в одну из триггерных ветвей. Он остается там и после снятия x и продолжает воздействовать на криотроны следующей ячейки. Это воздействие может быть передано в другую логическую схему через любой промежуток времени подачей в нужный момент импульса I_1 . Такого рода задержки удобны в сочетании с "двойной логикой".

7. Реализация автомата по его каноническим уравнениям

Пусть автомат задан своими каноническими уравнениями:

$$z_j(t) = f(x_1, x_2, \dots, x_m, q_1, q_2, \dots, q_k); \quad j = 1, 2, \dots, p;$$

$$q_i(t+1) = \varphi(x_1, x_2, \dots, x_m, q_1, q_2, \dots, q_k); \quad i = 1, 2, \dots, p. \quad (10)$$

Самым простым приемом реализации этих уравнений в вычислительной среде является отдельная реализация каждой из заданных функций. При этом функции $q_1(t+1)$ следует реализовать по принципу двойной логики с тем, чтобы $q_1(t+1)$ и $\bar{q}_1(t+1)$ через ячейку памяти типа триггера (рис. 9) передавались на входы $q_1(t)$ и \bar{q}_1 . Блоки, соответствующие функциям $z_j(t)$ и $q_1(t+1)$, следует располагать вертикально один под другим, если они имеют одинаковые переменные. Если среди функций $z_j(t)$ или $q_1(t+1)$ есть группы, не имеющие одинаковых переменных, то их можно расположить рядом, оставив между ними столбцы для подвода питания.

Схема, реализующая автомат с p выходами и k состояниями, изображена на рис. 10. Токи питания I_0^z для блоков $A(z_j)$, а также I_0^1 для ячеек памяти — постоянные. Токи питания блоков $A(q_1)$ и $\bar{A}(q_1)$ подаются импульсами в конце каждого такта. Длительность этих импульсов должна быть достаточной для полного завершения переходных процессов в блоках $A(q_1)$ и изменения состояний триггерных ячеек памяти.

Пример.

Канонические уравнения сумматора последовательного действия имеют следующий вид:

$$\left. \begin{aligned} z(t) &= \bar{x}_1 \bar{x}_2 q \vee x_1 x_2 q \vee \bar{x}_1 x_2 \bar{q} \vee x_1 \bar{x}_2 \bar{q} \\ q(t+1) &= x_1 x_2 \vee q x_1 \vee q x_2 \\ \bar{z}(t) &= \bar{x}_1 \bar{x}_2 \bar{q} \vee \bar{x}_1 x_2 q \vee x_1 \bar{x}_2 q \vee x_1 x_2 \bar{q} \\ \bar{q}(t+1) &= \bar{x}_1 \bar{x}_2 \vee \bar{q} \bar{x}_1 \vee \bar{q} \bar{x}_2 \end{aligned} \right\} \quad (II)$$

На рис. 11 показана программа, в которой блоки, реализующие каждую из четырех функций (II), расположены друг под другом.

Более компактные схемы можно получить, если использовать такую форму записи канонических уравнений, когда в некоторых из них выделены одинаковые члены (подфункции). Эти подфункции можно реализовать отдельными подсхемами и затем, используя каскадную реализацию, подавать результат на входы других логических функций. Иными словами, с целью упрощения программы можно использовать методы реализации многовыходных функций.

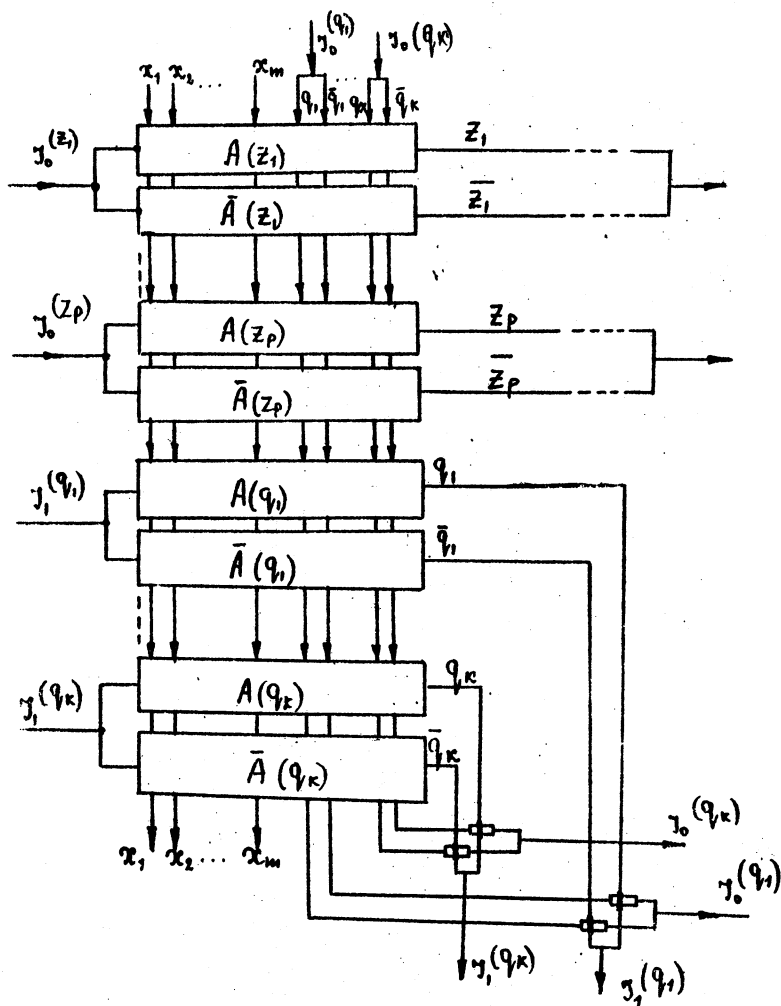


Рис. 10.

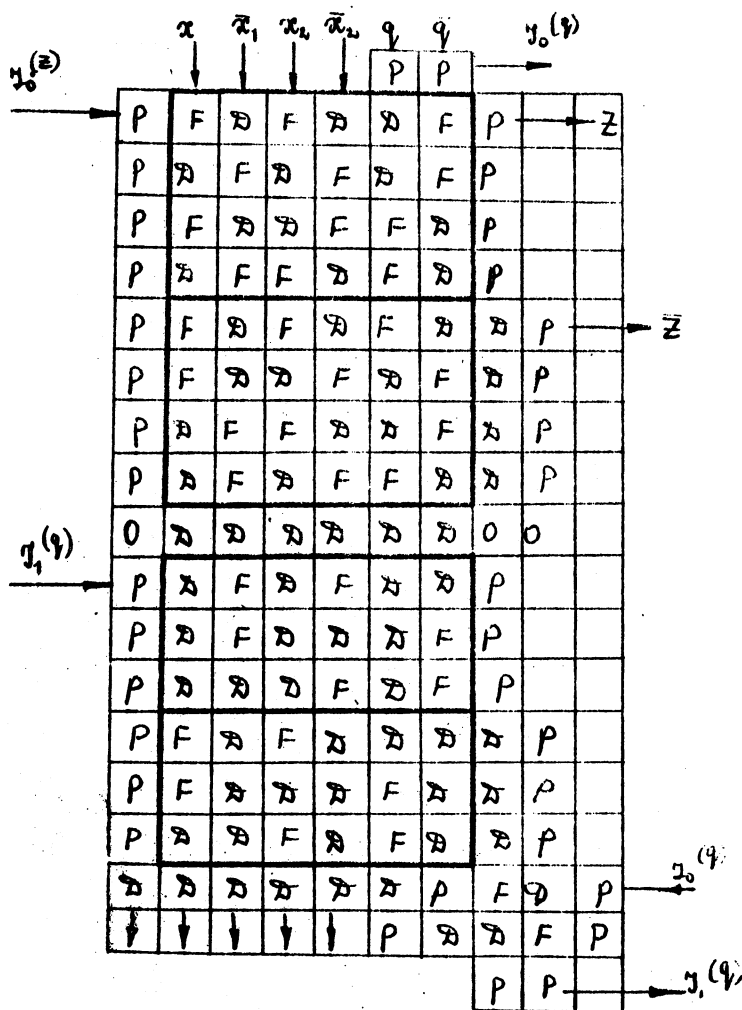


Рис. II

Для приведенного выше сумматора последовательного действия уравнения можно записать в следующей форме:

$$\left. \begin{aligned} q(t+1) &= x_1 x_2 \vee \bar{q} x_1 \vee q x_2, \\ z(t) &= \bar{q}(t+1)(q \vee x_1 \vee x_2) \vee q x_1 x_2, \\ \bar{q}(t+1) &= \bar{x}_1 \bar{x}_2 \vee \bar{q} \bar{x}_1 \vee q \bar{x}_2, \\ \bar{z}(t) &= q(t+1)(\bar{q} \vee \bar{x}_1 \vee \bar{x}_2) \vee \bar{q} \bar{x}_1 \bar{x}_2. \end{aligned} \right\} \quad (I2)$$

Из уравнений видно, что $z(t)$ и $\bar{z}(t)$, а также $q(t+1)$ и $\bar{q}(t+1)$ не имеют одинаковых переменных. Следовательно, имеет смысл располагать их блоки рядом. При этом питание приходится подводить сверху, разделяя блоки $A(q)$ и $\bar{A}(q)$ и также $A(z)$ и $\bar{A}(z)$ двумя столбцами. Выходы $q(t+1)$ и $\bar{q}(t+1)$ являются входами к расположенным снизу блокам $A(z)$ и $\bar{A}(z)$. Ячейка памяти занимает еще две строки под программой (рис. I2).

8. Реализация автомата по его канонической таблице

Для реализации некоторых простых автоматов можно применить каскадную реализацию, позволяющую построить программу непосредственно по заданной канонической таблице. Такой способ особенно удобен для автоматов с малым числом выходов и состояний и многими переменными.

Рассмотрим случай построения программы для автомата с $p = 1$ и $k = 1$ (один выход и одно состояние). Множество строк его канонической таблицы S можно разбить на 4 непересекающихся подмножества S_1, S_2, S_3 и S_4 следующим образом.

$$\left. \begin{aligned} s_k \in S_1, & \text{ если } f_1 = z(t)q(t+1) = 1, \\ s_k \in S_2, & \text{ если } f_2 = \bar{z}(t)\bar{q}(t+1) = 1, \\ s_k \in S_3, & \text{ если } f_3 = z(t)\bar{q}(t+1) = 1, \\ s_k \in S_4, & \text{ если } f_4 = \bar{z}(t)q(t+1) = 1. \end{aligned} \right\} \quad (I3)$$

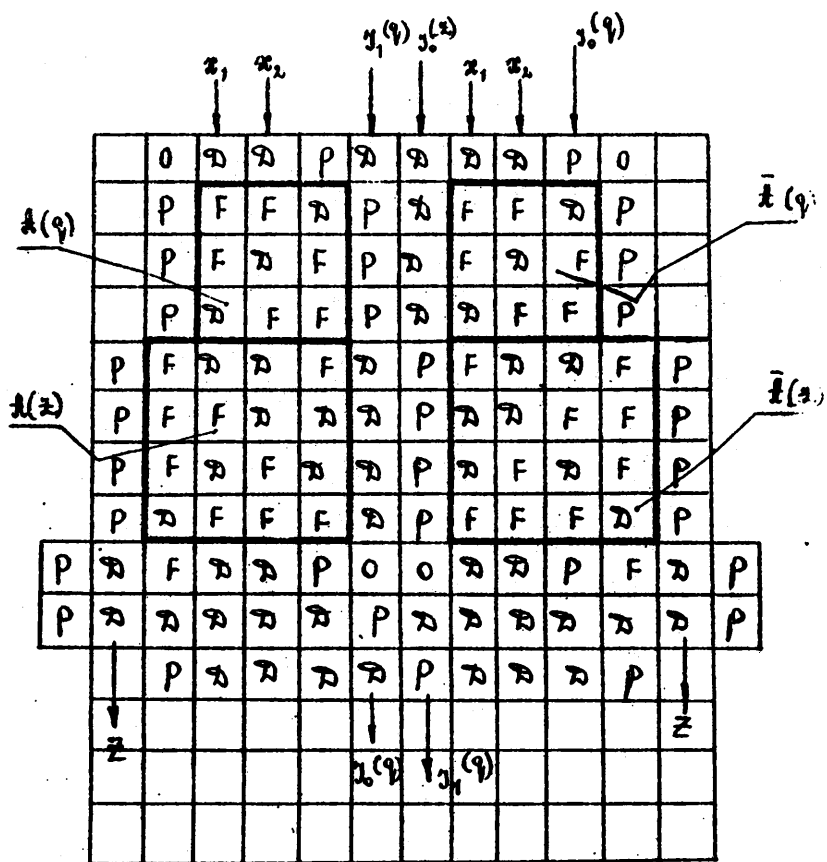


Рис. 12



2

Функции f_1, f_2, f_3 и f_4 могут служить подфункциями для каскадной реализации $z(t)$ и $q(t+1)$, поскольку (13) следует

$$\begin{aligned} z(t) &= f_1 \vee f_3, & q(t+1) &= f_1 \vee f_4, \\ \bar{z}(t) &= f_2 \vee f_4, & \bar{q}(t+1) &= f_2 \vee f_3. \end{aligned} \quad (14)$$

Программа, реализующая f_1, f_2, f_3 и f_4 (I-ый каскад) может строиться непосредственно по таблице. Для этого часть таблицы, соответствующая стандартным наборам переменных, переписывается с удвоенным числом столбцов (для $x_1, \bar{x}_1, x_2, \bar{x}_2, \dots, q_1, \bar{q}_1, \dots, q_n, \bar{q}_n$) и завершается четырьмя столбцами, соответствующими функциям f_1, f_2, f_3 и f_4 , в которых единицы расставляются в соответствии с уравнениями (13). Переход от таблицы к программе, реализующей функции f_1, f_2, f_3 и f_4 , осуществляется заменой всех нулей на элементы "D" единиц в стандартных наборах - на элементы "F" и единиц столбцах f_1, f_2, f_3 и f_4 - на элементы "P".

Реализация второго каскада чрезвычайно проста, т.е. для каждой функции (14) требуется только по одной строке. Выходы $q(t+1)$ и $\bar{q}(t+1)$ второго каскада через ячейку задержки соединяются с входами $q(t)$ и $\bar{q}(t)$ первого каскада.

На рис. 13 изображена программа, построенная по канонической таблице последовательного сумматора, заданного уравнениями (12) и рассмотренного в примерах.

Л и т е р а т у р а

1. Э.В. Еврешинов, Ю.Г. Косарев. Однородные вычислительные системы высокой производительности. Новосибирск, 1966 г.
2. Дж. Бремер. Сверхпроводящие устройства. М., 1964 г.
3. О.Л. Бандман, В.Ф. Гурко, Н.И. Назаров, Б.И. Фомель. Критические элементы вычислительной среды. Вычислительные системы. Труды Всесоюзной конференции по вычислительным системам, Новосибирск, 1967 г.
4. К. Шеннон. Синтез двухпольных переключавших схем. Работы по теории информации и кибернетике. М., 1963.
5. А.А. Кофман, В.А. Скоробогатов. Программирование для вычислительной среды. - Сб. "Вычислительные системы", вып. 26, Новосибирск, 1967.