В Ы Ч И С Л И Т Е Л Ь Н Ы Е С И С Т Е М Ы

Сборник трудов
1965 г. Института математики СО АН СССР Выпуск I8

НЕКОТОРЫЕ АЛГОРИТМЫ ПРОЕКТИРОВАНИЯ ЦВМ

Я.И. Фет

При построении принципиальной схемы ЦВМ разработчик выполняет много однообразных и сравнительно несложных операций, которые желательно передать машине. Особенно важное значение проблема автоматизации проектирования приобретает при разработке высокопроизводительных средств вычислительной техники [I]. Настоящая работа представляет собой попытку использовать ЦВМ для автоматизации перехода от функциональной схемы к принципиальной.

В § I рассматривается вопрос о представлении функциональной схемы ЦВМ в виде системы логических уравнений. В § 2 описан алгоритм логического моделирования функциональных схем. В § 3 изложены алгоритмы, которые позволяют, исходя из системы логических уравнений, описывающих данную функциональную схему, построить принципиальную схему для диодно-транзисторных элементов определенного типа. Предлагается способ выбора системы стандартных элементов, основанный на классификации логических функций по структуре их дизъюнктивных нормальных форм.

§ I

Ф у н к ц и о н а л ь н о й с х е м о й некоторого блока ЦВМ мы будем называть документ, в котором отражены все логи-

ческие и запоминающие элементы данного блока, функции элементов и связи между ними. Функциональная схема машины в целом задана, если известны функциональные схемы всех ее блоков и связи между блоками. П р и н ц и п и а л ь н а я с х е м а блока (или машины) – документ, отражающий реализацию данной функциональной схемы с помощью заданного набора физических элементов (запоминающих, логических и усилительных).

Функциональная схема любого реального устройства, перерабатывающего дискретную информацию, может быть задана в виде системы канонических уравнений [2]:

$$\begin{aligned} z_1(t) &= \varphi_1[x_1(t),\dots,x_m(t),q_1(t),\dots,q_k(t)],\\ &\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\ z_n(t) &= \varphi_n[x_1(t),\dots,x_m(t),q_1(t),\dots,q_k(t)],\\ q_k(t+1) &= \psi_1[x_1(t),\dots,x_m(t),q_1(t),\dots,q_k(t)],\\ &\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\ q_k(t+1) &= \psi_k[x_1(t),\dots,x_m(t),q_1(t),\dots,q_k(t)], \end{aligned} \qquad (I)$$

где $x_1(t),\dots,x_m(t)$ – значения логических переменных, сопоставленных сигналам на входах блока $x_1,\dots,x_m$ в момент времени t, $z_1(t),\dots,z_n(t)$ – то же на выходах блока, $q_1(t),\dots,q_k(t)$ – состояния запоминающих элементов блока, $\varphi_1,\dots,\varphi_n$ – функции выходов блока, $\psi_1,\dots,\psi_k$ – функции переходов блока. В случае двоичного кодирования, который нас интересует, все φ и ψ – функции алгебры логики.

Будем считать, что все уравнения системы приведены к ДНФ. В дальнейшем система логических уравнений, выраженных в ДНФ, обозначается сокращенно СЛУ. Условимся, кроме того, вместо $x_i(t), z_i(t), q_i(t), q_i(t+1)$ писать x_i, z_i, q_i, q_i', соответственно.

Если в качестве запоминающего элемента используется простая задержка, то СЛУ (I) может непосредственно служить исходной информацией для построения принципиальной схемы. Но в большинстве случаев запоминающие элементы выполняют и некоторые логические функции. К таким элементам относятся, например, различные триггеры. Работа такого элемента может быть описана уравнением

$$q' = f(\rho_1,\dots,\rho_s,q), \qquad (2)$$

где $\rho_1, \ldots, \rho_S$ – значения логических переменных, сопоставленных сигналам на входах элемента.

Чтобы реализовать с помощью этого элемента некоторую функцию перехода

$$q_i' = \psi_i(x_1, \ldots, x_m, q_1, \ldots, q_K), \qquad (3)$$

необходимо вычислить значения $\rho_1, \ldots, \rho_S$, решив совместно уравнения (2) и (3). Для этого можно воспользоваться, например, приведенными в [3] методами. В результате каждая функция перехода из СЛУ (I) представляется в виде одной функции типа (2) для соответствующего запоминающего элемента и S функций для входов этого элемента. СЛУ приобретает следующий вид:

$$\begin{aligned}
z_1 &= \Phi_1(x_1, \ldots, x_m, q_1, \ldots, q_K),\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
z_n &= \Phi_n(x_1, \ldots, x_m, q_1, \ldots, q_K),\\
\rho_{11} &= \psi_{11}(x_1, \ldots, x_m, q_1, \ldots, q_K),\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
\rho_{S1} &= \psi_{S1}(x_1, \ldots, x_m, q_1, \ldots, q_K),\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
\rho_{1K} &= \psi_{1K}(x_1, \ldots, x_m, q_1, \ldots, q_K),\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
\rho_{SK} &= \psi_{SK}(x_1, \ldots, x_m, q_1, \ldots, q_K),\\
q_1' &= f(\rho_{11}, \ldots, \rho_{S1}, q_1),\\
&\cdots\cdots\cdots\cdots\cdots\\
q_K' &= f(\rho_{1K}, \ldots, \rho_{SK}, q_K).
\end{aligned} \qquad (4)$$

На практике могут встретиться случаи, когда функциональная схема задана в обычной форме (чертеж с условными обозначениями). Тогда система логических уравнений должна быть построена с помощью алгоритма анализа схемы, описанного в [2]. Однако в таких случаях необязательно приводить систему уравнений к канонической форме. Программы автоматического проектирования применимы и к СЛУ, представленным в следующей форме:

$$\begin{aligned}
y_1 &= \varphi_1(x, q),\\
y_2 &= \varphi_2(x, q, y_1),\\
&\cdots\cdots\cdots\cdots\\
y_\ell &= \varphi_\ell(x, q, y_{\ell-1}),\\
z &= \Phi(x, q, y),\\
q' &= \Psi(x, q, y),
\end{aligned} \qquad (5)$$

где $y_1, \ldots, y_\ell$ – промежуточные переменные, расположенные в порядке возрастания рангов соответствующих подсхем схемы.

Итак, можно считать, что функциональная схема ЦВМ задана в виде СЛУ. Все уравнения СЛУ желательно представить в такой форме, чтобы обеспечить экономную реализацию. Среди работ, посвященных вопросам минимизации функций алгебры логики и СЛУ отметим статьи [4], [5], [6], [7] и [8].

В данной работе вопросы минимизации не рассматриваются. Впрочем, применение описываемых здесь алгоритмов проектирования допускает любые преобразования исходной СЛУ. Единственное требование состоит в том, чтобы каждое уравнение было представлено в ДНФ.

§ 2

Прежде чем приступить к построению принципиальной схемы, целесообразно проверить, правильно ли составлена функциональная схема. Иными словами, нужно определить, действительно ли данная СЛУ описывает такую переработку информации, какую должен выполнять проектируемый блок. Разумеется, при том большом числе внутренних состояний, которым обладает всякий реальный блок ЦВМ, практически невозможно проделать полный анализ. Однако можно получить приемлемые результаты с помощью логического моделирования. Логическое моделирование состоит в многократном потактном решении СЛУ с последующим сравнением полученных значений выходных переменных с ожидаемыми результатами. Обычно достаточно проделать несколько десятков или сотен циклов логического моделирования над разными комбинациями входных переменных. Эти комбинации выбирают так, чтобы охватить все ситуации, которые представляются "опасными". Таким образом, логическое моделирование сводится в основном к решению большого числа логических уравнений и легко может быть автоматизировано.

На рис. I приведена блок-схема программы логического моделирования. В массив исходных данных, кроме СЛУ, входит последовательность значений входных переменных, показатель числа тактов и ожидаемая последовательность выходных переменных. При кодировании логических уравнений используется следующий принцип: имя каждой логической переменной совпадает с адресом ячейки оперативного запоминающего устройства (ОЗУ), в которой хранится текущее значение этой переменной.[I] Для переменных отводится специальное поле в ОЗУ, которое делится на 5 частей: район z, район y, район q', район x и район q. Так, если емкость ОЗУ моделирующей ЦВМ равна 2^{12}, то можно распределить память следующим образом:

0000 - 0777[2] программа моделирования, рабочие ячейки и т.д.

I000 - I777 район z
2000 - 2777 район y
3000 - 3777 район q'
4000 - 4777 район x
5000 - 5777 район q
6000 - 7777 СЛУ моделируемого блока.

Далее вводится еще ряд правил кодирования логических уравнений.

I. Знаки равенства и конъюнкции опускаются.

2. Каждая переменная кодируется пятиразрядным восьмеричным кодом, причем младшие четыре разряда составляют имя переменной, а в старшем разряде содержится дополнительная информация (см. п. 3).

Примечание: для переменных y, находящихся в правой части уравнений, в четвертом разряде кода вместо 2 следует ставить 6.

3. В старшем разряде кода переменной двоичные единицы служат признаками следующих операций:

5-й разряд

		I
	I	
I		

переменная входит в уравнение с отрицанием;
переменная завершает элементарное произведение;
переменная завершает уравнение.

I) Такой способ кодирования предложен Стокуэллом в работе [9].
2) Нумерация ячеек восьмеричная.

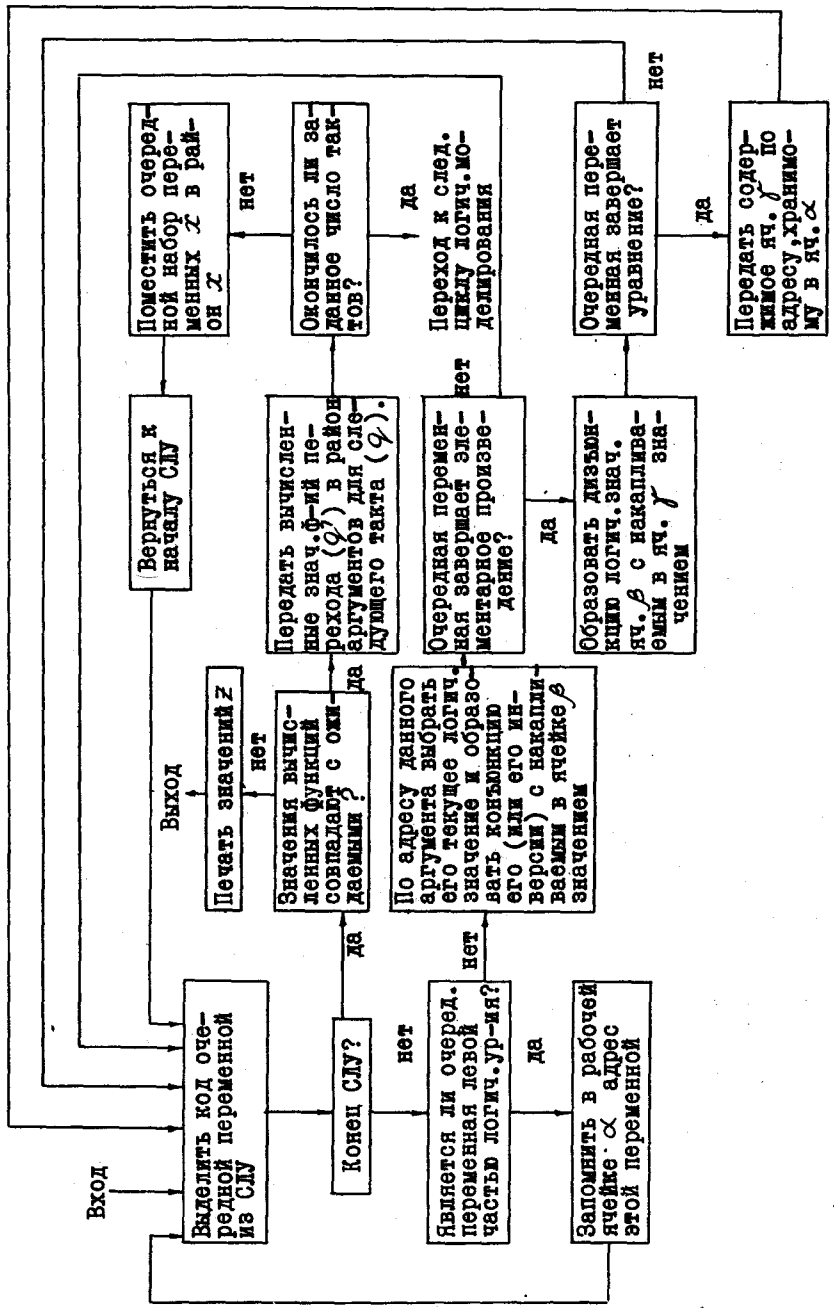

Рис. I. Блок-схема программы логического моделирования.

4. Код первой переменной каждого уравнения непосредственно примыкает к коду последней переменной предыдущего уравнения.

П р и м е р ы

Уравнения:

$$z_2 = x_1 \bar{x}_4 q_2 \vee x_2 \bar{q}_1 q_5 \qquad (6)$$

$$q'_{15} = x_3 \bar{q}_{10} \vee \bar{x}_{127}$$

Коды:

```
0I002   04001   I4004   25002   04002   I5001   65005
030I7   04003   350I2   74I77                            (7)
```

Размещение кодов в 45-разрядных ячейках ОЗУ:

```
0I002   04001   I4004
25002   04002   I5001
65005   030I7   04003
350I2   74I77   00000
```

Отметим, что изложенные правила легко позволяют составить программу автоматического кодирования СЛУ. Однако использование такой программы целесообразно только в том случае, когда имеется читающее устройство, позволяющее вводить логические уравнения в обычной записи (6). Если же для подготовки к вводу приходится вручную превращать обычные записи в какой-либо цифровой код, то проще непосредственно записывать коды (7).

Для программирования оказывается удобным хранить логическое значение переменной одновременно во всех разрядах соответствующей ячейки. Так, если в данном такте x_1 = I, а x_2 = 0, то во всех разрядах ячейки 400I стоят единицы, а ячейки 4002 – нули.

Программа моделирования в каждом такте последовательно просматривает всю СЛУ и для каждой функции y, z и q' вычисляет ее значение. При этом в качестве аргументов используются текущие значения x, y и q. Для каждого элементарного произведения постепенно вычисляется логическое произведение всех его переменных. По окончании элементарного произведения полученная величина логически суммируется с накопленной для данного уравнения суммой. Вычисленные значения y, z и q' заносятся в соответствующие районы. При переходе к следующему такту программа заносит в район "x" очередной набор входных

сигналов, а в район "q" – содержимое района "q'" (разность между именами q_i и q'_i – постоянное число, в нашем случае 2000).

Выдача результатов может быть организована по-разному. В приведенной блок-схеме, например, печатаются значения всех z в том случае, если они не совпадают полностью с ожидаемыми. На основании этих данных вносятся исправления в исходную систему логических уравнений.

§ 3

В отличие от "абстрактного" подхода, допустимого для логического моделирования, при построении принципиальной схемы необходимо учитывать физические особенности используемых элементов и сигналов. Задача построения н а д е ж н о й схемы ЦВМ будет здесь решаться лишь в первом приближении, а именно, будет учитываться нагрузочная способность элементов. Некоторые физические особенности схемы (например, влияние соединительных проводов) могут быть выявлены после размещения элементов и составления монтажной схемы. Полную гарантию работоспособности ЦВМ обычно дает только отладка ее образца.

Построение принципиальной схемы мы будем проводить для случая, который характеризуется следующими условиями:

I. Комбинационная сеть реализуется суперпозициями двухкаскадных схем.

2. Каждый логический элемент имеет структуру, показанную на рис. 2. Разные элементы отличаются числом входов конъюнкторов и дизъюнкторов.

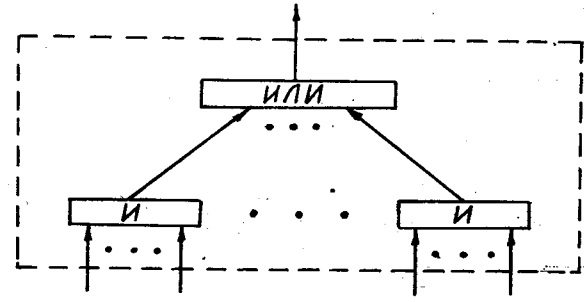

Рис. 2. Логическая структура элемента.

3. На выходах запоминающих и логических элементов одновременно вырабатываются функции этих элементов и их инверсии.

4. Одновременно с каждой входной переменной существует ее инверсия.

5. Нагрузочная способность источников входных переменных не ограничена.

Программа построения принципиальной схемы делится на четыре этапа: I) реализация, 2) построение принципиальной таблицы, 3) проверка нагрузок, 4) составление таблицы соединений.

В общих чертах работа программы на этих этапах заключается в следующем.

Р е а л и з а ц и я. Для каждой функции из СЛУ выбирается логический элемент, который будет реализовать эту функцию. При этом принимаются меры, обеспечивающие экономную реализацию.

П о с т р о е н и е п р и н ц и п и а л ь н о й т а б л и ц ы. В соответствии с принятой на предыдущем этапе реализацией за каждой переменной СЛУ закрепляется конкретный контакт конкретного стандартного элемента.

П р о в е р к а н а г р у з о к. Исходя из полученных в принципиальной таблице связей и заданных нагрузочных характеристик элементов, производится проверка допустимых нагрузок и, в случае необходимости, исправление принципиальной таблицы.

С о с т а в л е н и е т а б л и ц ы с о е д и н е н и й. Пользуясь исправленной принципиальной таблицей, программа составляет таблицу соединений, в которой для каждого контакта схемы указывается с какими другими контактами он соединен.

РЕАЛИЗАЦИЯ

С целью систематизации логических функций и упрощения синтеза схем из описанных выше элементов целесообразно ввести некоторую классификацию ДНФ функций алгебры логики.

Число логических переменных, входящих в элементарное произведение, будем называть р а н г о м к о н ъ ю н к ц и и (r_K). Число элементарных произведений, входящих в данную ДНФ - р а н г о м д и з ъ ю н к ц и и (r_g). Каждой ДНФ сопоста-

вим с и м в о л - целое число $r_{K_1} r_{K_2} \dots r_{K_{r_g}}$, где $r_{K_1}, r_{K_2}, \dots$ - ранги конъюнкций I-го, 2-го, ... элементарных произведений, а r_g - ранг дизъюнкции данной ДНФ.

Символ называется у п о р я д о ч е н н ы м, если все r_K символа расположены в монотонно невозрастающей последовательности.

Пример:

Формуле $x_1 \bar{x}_2 x_3 \vee x_2 x_3 \bar{x}_4 \bar{x}_5 \vee \bar{x}_3$ соответствует символ 34I (упорядоченный символ 43I).

Каждому логическому элементу описанного выше типа может быть сопоставлен аналогичный с и м в о л э л е м е н т а, характеризующий его логические возможности.

Две ДНФ формул алгебры логики относятся к одному и тому же семейству K -го рода, если их упорядоченные символы имеют одинаковые r_g и при поразрядном вычитании меньшего символа из большего разность в каждом разряде не превосходит K.

Всякая СЛУ может быть разбита на семейства 0,I и т.д. родов. Все формулы, входящие в некоторое семейство K -го рода, могут быть реализованы стандартным элементом, символ которого равен максимальному символу данного семейства, причем в каждом конъюнкторе этого элемента остается не более K неиспользуемых входов.

С ростом K количество семейств в разбиении данной СЛУ (а, следовательно, и количество различных стандартных элементов) уменьшается. В предельном случае, при $K=(r_K)_{max}$ (для данного r_g) все формулы с данным r_g реализуются одним стандартным элементом.

Однако с ростом K увеличивается число неиспользуемых входов элементов, т.е. общая стоимость реализации СЛУ.

Каждое из разбиений (0,I,..., K -го рода) можно рассматривать как один из вариантов реализации. Программа анализа вариантов реализации (АНР) осуществляет эти разбиения и вычисляет условную стоимость реализации для каждого варианта.

Условная стоимость

$$c_j = \sum_S c_S \cdot k_f^S ,$$

где c_j - общая условная стоимость реализации СЛУ для j -го разбиения.

k_f^S - количество функций, реализуемых стандартным элементом с символом S.

c_S - условная стоимость элемента с символом S.

Суммирование производится по всем символам стандартных элементов j -го разбиения.

Величина C_S для каждого стандартного элемента подсчитывается по формуле

$$C_S = \sum_{i=1}^{r_g} r_{x_i} + r_g + const.$$

Первые два слагаемых в правой части этой формулы дают общее число диодов во входной логике стандартного элемента, третье слагаемое – константа, которая зависит от схемы и конструктивного выполнения стандартного элемента и задается разработчиком.

Исходными данными для программы АВР служат СЛУ заданной функциональной схемы и некоторые константы, задаваемые разработчиком.

Результат работы программы: набор таблиц реализации (ТР) для семейств 0,I,...,K -го рода.

На рис. 3 приведена форма таблицы реализации.

Символ стандартного элемента S_i	Условная стоимость этого стандартного элемента C_{S_i}	Количество ф-ий, реализуемых им $K_f^{S_i}$	Имена реализуемых функций

Основные характеристики:

I. Род разбиения j .

2. Количество семейств K_j .

3. Общая условная стоимость реализации C_j .

Рис. 3. Таблица реализации.

На рис. 4 приведена блок-схема программы АВР.

Разработчик выбирает наиболее выгодный из вариантов реализации и составляет для него таблицу стандартных элементов (ТСЭ). Введем некоторые определения, необходимые для описания стандартных элементов.

В е с в х о д а. Минимальная из входных проводимостей входов всех элементов принимается за единицу веса. Тогда вес любого входа каждого элемента может быть приблизительно выражен в единицах веса.

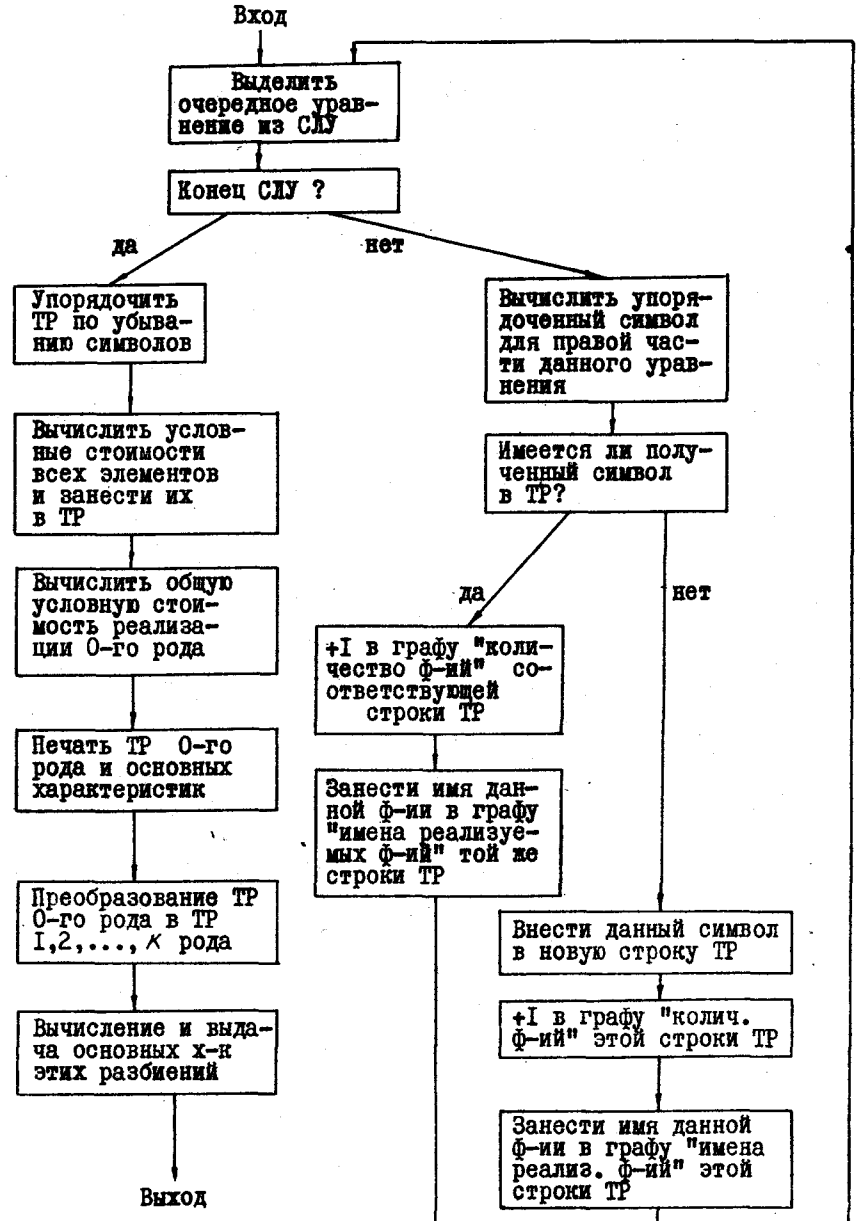

Рис. 4. Блок-схема программы АВР.

В е с в ы х о д а элемента характеризует нагрузочную способность и измеряется числом входов единичного веса, которые могут быть одновременно подключены к данному выходу.

А д р е с в х о д а (выхода) - номер контакта на плате стандартного элемента или на разъеме ячейки, на который подается (или с которого снимается) соответствующий сигнал.

ТСЭ содержит следующую информацию о каждой ячейке:

1. Символ стандартного элемента.
2. Количество стандартных элементов в ячейке.
3. Адрес и вес выхода.
4. Адрес и вес инверсного выхода.
5. Адреса и веса всех входов (перечисляются в порядке, соответствующем символу элемента).

П р и м е ч а н и я : 1) если в ячейке содержится более одного стандартного элемента, то информация по п.п. 3, 4, 5 повторяется последовательно для каждого из стандартных элементов. 2) В конце ТСЭ приводится в аналогичной форме информация о запоминающих и усилительных элементах.

Выбранная ТР, ТСЭ и СЛУ составляют исходные данные для работы программы построения принципиальной таблицы (ППТ).

В некоторых случаях набор логических элементов задается независимо. Тогда применение программы АВР необязательно. Разработчик должен составить ТСЭ для заданного набора в такой же форме, как показано выше. Эта таблица и СЛУ служат исходными данными для программы реализации (ПР). Работа заключается в следующем:

1. Составляется таблица реализации 0-го рода.
2. Символы элементов из ТСЭ упорядочиваются по неубыванию вычисленных программой условных стоимостей элементов.
3. Для каждого символа формулы из таблицы реализации 0-го рода находится первый по порядку (т.е. с наименьшей стоимостью) стандартный элемент, реализующий данный символ формулы.

Символ этого элемента заносится в графу "символ стандартного элемента" соответствующей строки таблицы реализации 0-го рода. Полученная после реализации всех символов ТР входит в массив данных для ППТ.

На рис. 5 приведена блок-схема программы реализации.

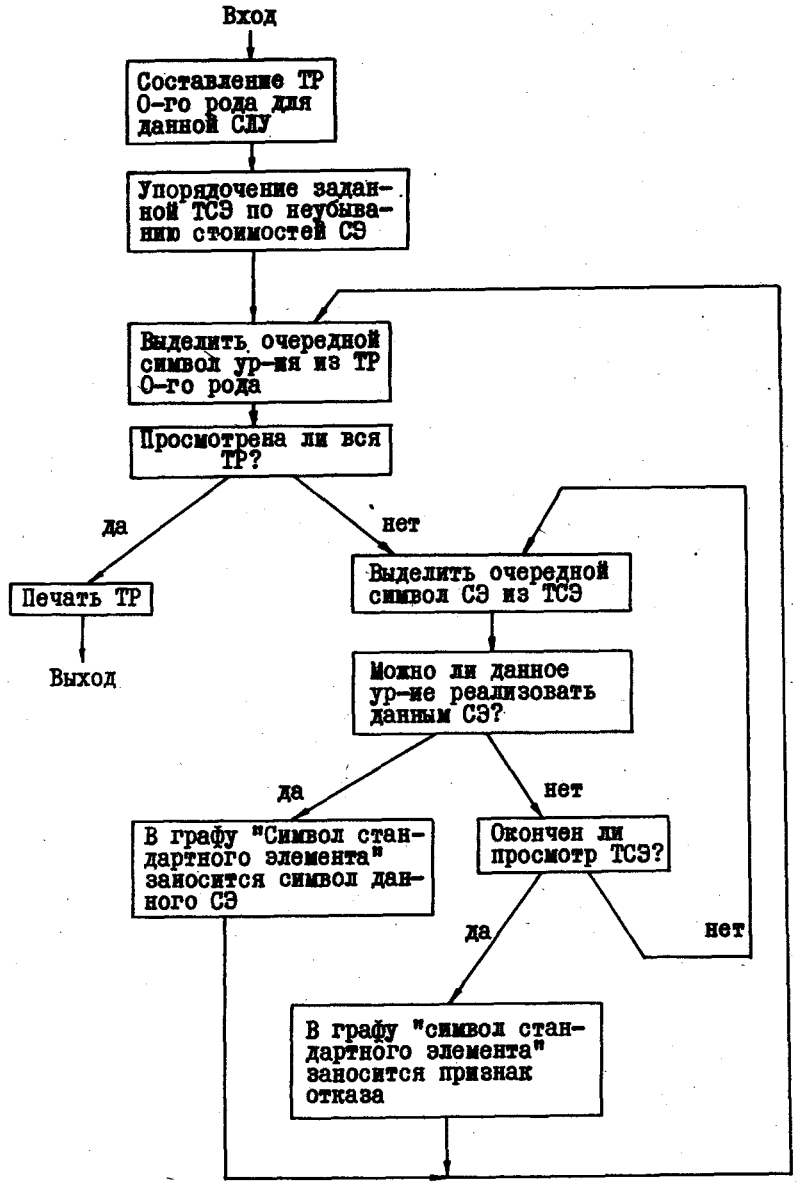

Рис. 5. Блок-схема программы реализации (ПР).

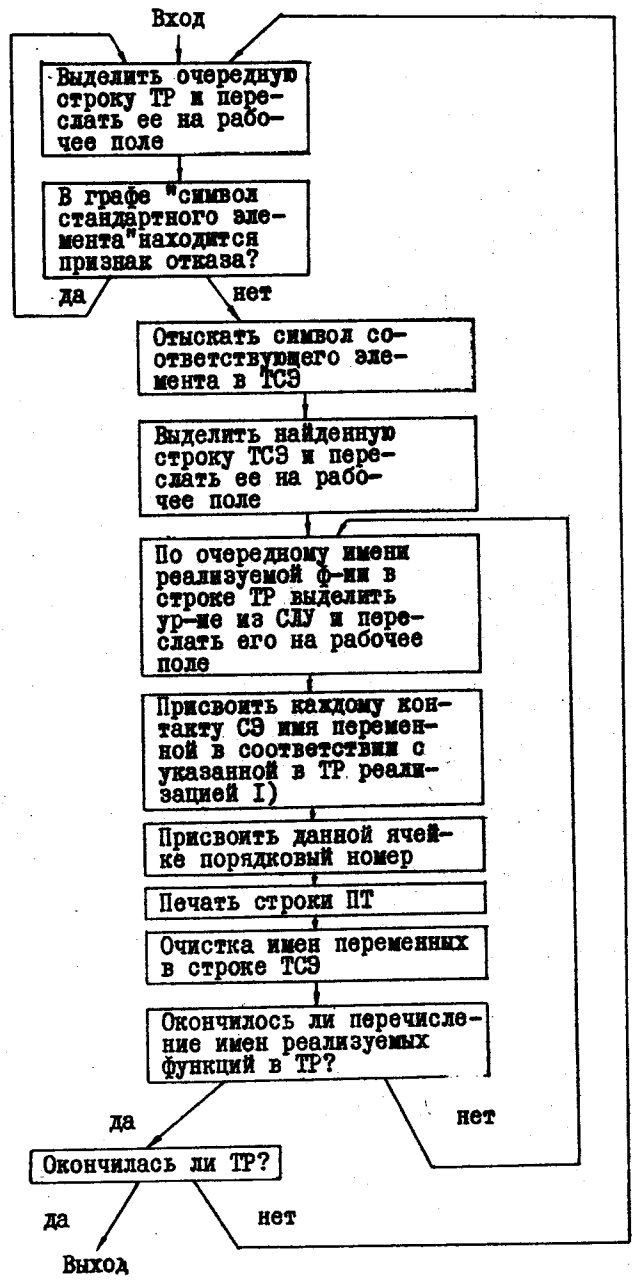

Рис. 6. Блок-схема программы ППТ.

I) В процессе присвоения в ячейки для переменных заносятся значения $W_{вых.}$ и нарастающие суммы $\Sigma W_{вх.}$ и $\Sigma W_{вх.}'$.

ПОСТРОЕНИЕ ПРИНЦИПИАЛЬНОЙ ТАБЛИЦЫ

Работа программы ППТ состоит в следующем:

I. Из ТР выделяется очередной символ стандартного элемента.

2. Из ТСЭ выделяется строка с информацией о ячейке с соответствующими стандартными элементами.

3. С помощью указанных в ТР имен функций, реализуемых данным стандартным элементом, из СЛУ выделяется логическое уравнение очередной реализуемой функции.

4. Имя каждой логической переменной рассматриваемого уравнения приписывается к соответствующему адресу выхода или входа в строке ТСЭ.

5. Веса заполняемых выходов или входов заносятся по адресам, соответствующим именам переменных.I)

6. Выделенной ячейке присваивается порядковый номер.

7. Дополненная указанным образом строка ТСЭ представляет собой строку принципиальной таблицы (ПТ) и выдается на внешние запоминающие устройства. После этого начинается обработка следующего логического уравнения (или, если все функции, реализуемые данным стандартным элементом, исчерпаны - следующего символа).

На рис. 6 приведена блок-схема программы ППТ.

На рис. 7 приведена форма принципиальной таблицы. В тех случаях, когда некоторые входы не используются, в соответствующей графе "Имя входной переменной" проставляются нулевые коды.

8. После обработки всех символов стандартных логических элементов процедура присвоения имен переменных и нумерации ячеек применяется к запоминающим элементам. При этом обрабатываются функции f из СЛУ (4); эти функции при кодировании отмечаются особым признаком.

ПРОВЕРКА НАГРУЗОК (ПН)

По окончании работы программы ППТ в районах y и y' находится информация о нагрузочной способности стандартных элементов, реализующих эти функции, и о действительной нагрузке этих элементов в схеме. Эта информация имеет следующую форму записи:

I) Эта операция подготавливает информацию для работы программы проверки нагрузок.

Символ (тип) стандартного элемента	Порядковый номер ячейки	Адрес выхода	Имя выходной переменной	Адрес I-го входа	Имя I-ой входной перем.	...	Адрес K-го входа	Имя K-ой входной переменной

Рис. 7. Принципиальная таблица.

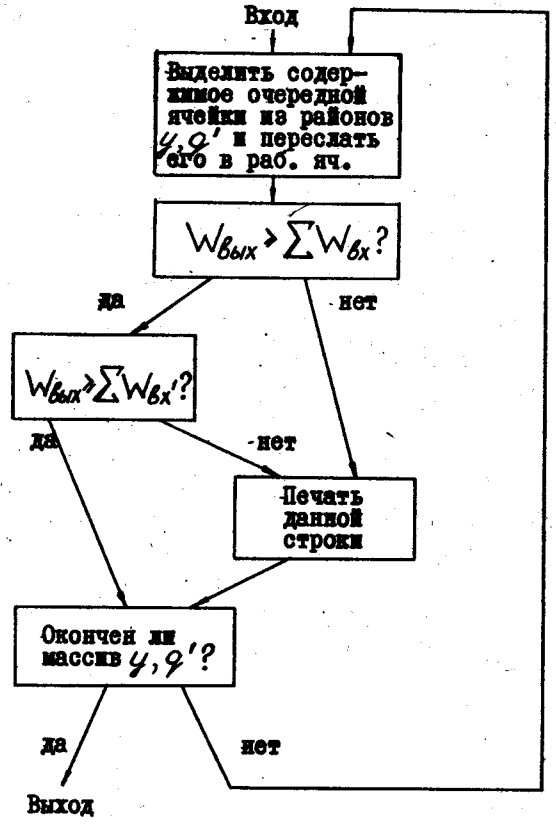

Рис. 8. Блок-схема программы ПН.

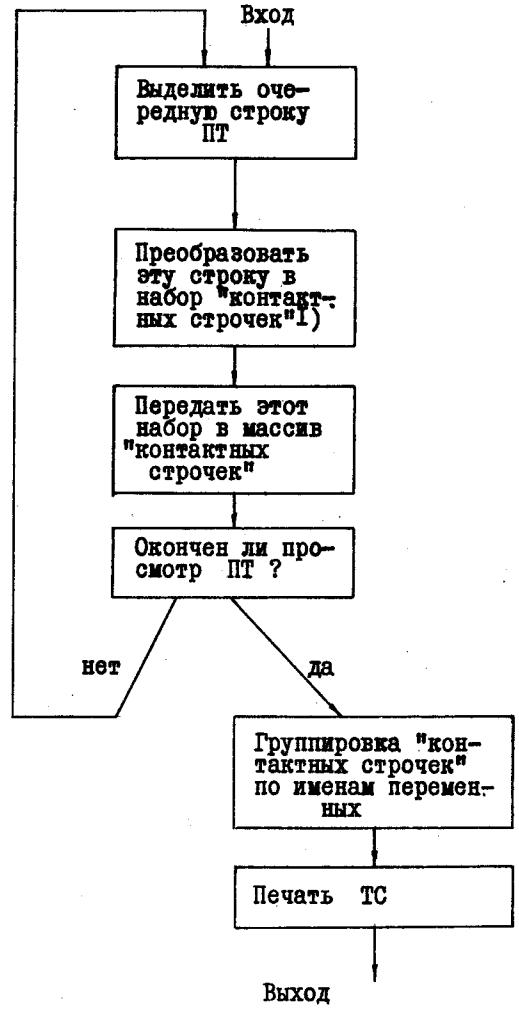

Рис. 9. Блок-схема программы СТС.

I) Форма "контактной строчки":

Тип СЭ	Порядковый номер ячейки	Адрес переменной	Имя переменной

Число "контактных строчек", соответствующих каждой строке ПТ, равно количеству используемых контактов данного элемента.

$W_{вых}$	$\sum W_{вх}$	$\sum W_{вх'}$

где $W_{вых}$ – вес выхода стандартного элемента,
$\sum W_{вх}$ – сумма весов всех входов схемы, соединенных с данным выходом,
$\sum W_{вх'}$ – то же для инверсного выхода.

Правило нагрузок выражается следующими неравенствами:

$$W_{вых} \geqslant \sum W_{вх}$$
$$W_{вых} \geqslant \sum W_{вх'} \qquad (8)$$

Программа ПН проверяет эти неравенства для всех функций y и y'.

Информация о функциях, для которых не выполняются неравенства (8) выдается на печать. Другой вариант заключается в том, что для выходов, на которых не выполняются неравенства (8), ставятся усилительные элементы из ТСЭ, затем вносятся исправления в ПТ и повторно проверяются условия (8).

СОСТАВЛЕНИЕ ТАБЛИЦЫ СОЕДИНЕНИЙ (СТС)

Блок-схема программы СТС приведена на рис. 9.

Исходной информацией для работы программы СТС является ПТ. Просматривая последовательно всю ПТ, программа объединяет в отдельные группы все адреса входов и выходов, на которых реализуется одна и та же переменная. Это осуществляется путем упорядочения массива "контактных строчек" по неубыванию кодов имен переменных.

Описанные программы позволяют получить ряд документов, которые могут быть использованы непосредственно, а также подвергнуты дальнейшей обработке на ЦВМ с целью составления монтажных схем и другой производственной и эксплуатационной документации.

В заключение автор выражает благодарность В.Э. Гейту и Ю.И. Колосовой, составившим программы по всем описанным в настоящей статье алгоритмам.

ЛИТЕРАТУРА

1. Евреинов Э.В., Косарев Ю.Г. О вычислительных системах высокой производительности. Изв. АН СССР, сер. технич.киберн., 1963,№4,3-23.
2. Кобринский Н.Е., Трахтенброт Б.А. Введение в теорию конечных автоматов. М., Физматгиз,1962.
3. Фистер М. Логическое проектирование цифровых вычислительных машин. Пер. с англ. под ред. В.М.Глушкова. Киев, "Техника", 1964.
4. Troye N.C.de. Classification and minimization of switching functions. Phillips Res.Repts.,1959, 14, N 2, 151-193, N 3,250-292.
5. Kudielka V. Ein Verfahren zur Ermittlung aller nicht redundanten zweistufigen Darstellungen einer logischen Funktion. Elektron.Rechenanl.,1963,5,N1,11-21.
6. Bartee T.C. Computer design of multiple-output logical networks. IRE Trans.Electron.Comput.,1961,EC-10,N1, 21-30.
7. Polansky R.B. Minimization of multiple-output switching circuits. Commun. and Electron.,1961, N 53,67-73.
8. Казаков В.Д. Минимизация логических функций большого числа переменных. Автоматика и телемеханика,1962, 23,№9, 1237-1242.
9. Stockwell G.N. Computer logic testing by simulation. IRE Trans.Mil.Electr., 1962, MIL-6, N 3, 275-282.

Ин-т математики СО АН СССР

Поступила в редакцию
I.II.1964 г.