ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ С ПРОГРАММИРУЕМОЙ СТРУКТУРОЙ
(Вычислительные системы)
1982 год Выпуск 94

УДК 681.3:62-52

ПОСТРОЕНИЕ ИНФОРМАЦИОННО-УПРАВЛЯЮЩИХ СИСТЕМ ДЛЯ АВТОМАТИЗАЦИИ СТЕНДОВЫХ ЭКСПЕРИМЕНТОВ

В.Г.Цыбинский

Стендовые эксперименты с использованием сложных установок "полупромышленного" типа получили сейчас широкое распространение как в области фундаментальных наук (физики, химии, биологии), так и в области испытаний объектов новой техники (например, вибробарокамеры). Для таких установок (стендов) характерно большое количество регулируемых параметров и замкнутых контуров регулирования. Стенды позволяют получить много информации, но требуют больших усилий по обеспечению их нормального функционирования. Зачастую без привлечения средств автоматизации их эксплуатация вообще невозможна (неэффективна). Традиционно для этих целей используются мини-ЭВМ, пришедшие на смену управляющим ЭВМ II поколения. Недостаточные успехи применения этих средств обусловлены, на наш взгляд, несоответствием их идеологии (концентрация вычислительных средств) основным требованиям к системам автоматизации стендов:

- быстрота настройки и перестройки (гибкость);
- живучесть;
- хорошее взаимодействие с объектом и экспериментатором (повышенный "интеллект").

Микропроцессорная техника позволяет перейти к системам распределенного управления, в значительно большей степени удовлетворяющим требованиям, изложенным выше. Статья посвящена одной из возможных реализаций таких систем в виде однородных вычислительных систем [I]. Сохраняя традиционным общее построение: активный элемент (процессор) → память → модули связи с объектом, такие системы достаточно эффективно удовлетворяют вышеупомянутым требованиям.

I. Гибкость системы (быстрота перестройки). Настройка (и перестройка) системы включает в себя комплекс монтажных и наладочных работ и работы по настройке программного аппарата, причем вторая часть работ является более трудоемкой. Повсеместно принято, что программы реального времени должны быть ассоциативными. Тогда настройка заключается в составлении баз данных, описывающих монтаж, и перекомпиляции программ. В дальнейшем будет показано, что компиляцию лучше заменить интерпретацией, производя подмену адресов непосредственно при выполнении программы. Сейчас рассмотрим затраты по составлению баз данных.

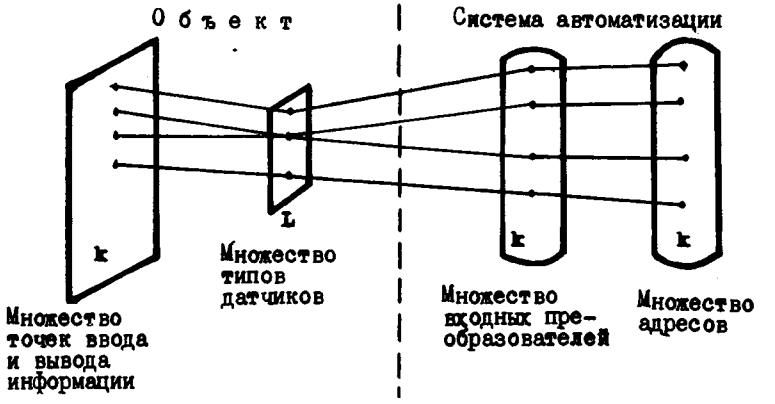


Рис. I

Подключение и настройка системы автоматизации к объекту включает в себя определение отношения на четырех множествах (рис.I):

- множество точек ввода и вывода информации на объекте (представляется, как правило, набором символических имен);
- множество типов датчиков;
- множество преобразователей системы, т.е. множество входов системы;
- множество адресов этих преобразователей, с которыми работают проблемные программы.

Первые два множества относят к объекту. Для определения этого отношения необходимо некоторое количество информации. Определим эти информационные затраты (в пересчете на один вектор, т.е. на

одну точку обмена информацией) и рассмотрим их связь со структурой системы.

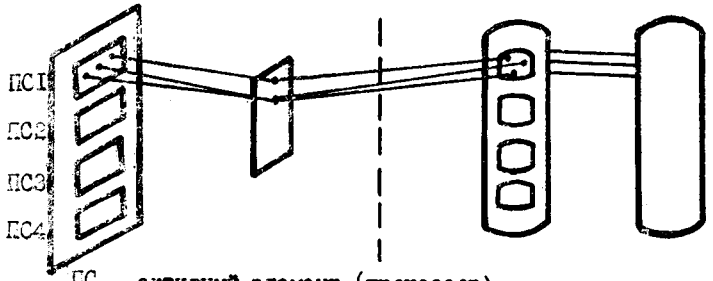
Таблица I

Тип схемы	Преобраз.-адреса	Датчики-преобразов.	Точка - тип датчика
Полная независимость	$\frac{1}{k}\log k!$		
Группировка по типам	$\frac{1}{k}\log\alpha!+M[\sum_{i}^{\alpha}\log k^i!]$		
Группировка с привязкой	$\frac{1}{k}M[\sum_{i}^{\alpha}\log k^i!]$	$\frac{1}{k}M[\sum_{i}^{\alpha}\log k^i!]$	$(1-\frac{\alpha}{k})\log\alpha$
Мультиплексирование	$\frac{1}{k}M[\sum_{i}^{\alpha}\log\frac{k^i}{\varphi_i}!]$		
Привязка всех адресов	- - - - - - - - -		
Представление подсистем	- - - - - - - - -	$\frac{1}{k}M[\sum_{i}^{\alpha}\sum_{j}^{n}\log k_j^i]$	

Количество информации определяется как логарифм числа равновероятных исходов, приведенных к числу точек k. Исходные формулы сведены в табл.I. Верхняя строка таблицы соответствует случаю, когда система не упорядочена, т.е. каждый преобразователь может иметь любой адрес (преобразователи объединены на основе полностью параллельного интерфейса и независимы). В последующих строках таблицы приведены оценки приемов структурирования системы, каждый из которых находит применение в существующих системах:

- группировка преобразователей по типам, при этом преобразователи одного типа занимают соседние места в адресном пространстве;
- дополнительная привязка начальных адресов групп к фиксированным адресам в адресном пространстве;
- мультиплексирование;

- полная привязка адресов к расположению преобразователей, т.е. использование радиального интерфейса. Эти приемы позволяют уменьшить количество информации, необходимое для отношения "преобразователи-адреса". Количество информации для отношения "точки на объекте-входы преобразователей" при этом, естественно, не меняется. Оно может быть уменьшено, если объекту (стенду), представленному совокупностью подсистем (что, как правило, имеет место на практике), сопоставлена система, имеющая такую же структуру



ПС - активный элемент (процессор)

Рис. 2

(рис.2). Таким образом, речь идет о некотором "информационном дублировании" стенда в системе, причем степень автономии подсистемы в объекте соответствует степени автономии ячейки (элемента) системы автоматизации, и т.д. Этому соответствует нижняя строка таблицы. Так как количество информации зависит от некоторых параметров объекта и точек обмена информацией, которые можно считать случайными (распределение датчиков по типам, количество подсистем) в формулы введен символ математического ожидания M. Другие параметры: α - число типов датчиков, n - число подсистем, φ - степень мультиплексирования.

Исследование приведенных в табл.I зависимостей показало уменьшение количества информации благодаря использованию приемов, указанных выше. Можно сказать, что мультиплексирование и привязка адресов, с одной стороны, и "информационное" дублирование стенда существенно уменьшают требуемое количество информации и увеличивают гибкость системы. Эти рассуждения учитываются при выборе структуры информационно-управляющей среды.

С точки зрения связей между подсистемами объект в общем случае представляет собой структуру типа ориентированного графа. Сис-

темы, имеющие такую структуру, вообще говоря, плохо организованы (в них сложно оформлять протоколы и т.д.). Как показывает анализ, большинство экспериментальных стендов имеют более простую структуру - типа дерева, где каждая подсистема имеет несколько обеспечивающих, и т.д. Иерархичность является естественным свойством сложных структур [2]. Так как ориентированный граф вообще может быть сведен к дереву введением некоторой избыточности [3], можно принять структуру информационно-управляющей системы иерархической, состоящей из одинаковых подсистем (ячеек), оставив возможность расширения к ориентированному графу. Ячейки должны обладать некоторым "начальным интеллектом" и иметь возможность заменять друг друга. В этом случае естественно организовать резервирование по принципу "верхний резервирует нижнего". Очевидно, что при таком построении системы каждый вышестоящий узел дерева может резервировать любой из нижестоящих. Может ли он при этом сам быть замещенным, т.е. какова глубина резервирования? Глубокое резервирование предполагает глубокое же разделение ресурсов, доступных ячейке, между уровнями. Величина адресного поля, количество внешних устройств, количество сигналов прерывания должны быть разделены между ячейками способными резервировать друг друга. В дальнейшем рассматривается вариант с резервированием на глубину, равную I. Это позволяет разделить ресурсы между двумя соседними уровнями и в то же время обеспечивает достаточно высокий уровень надежности (живучести).

2. Надежность. Определим возрастание надежности системы при введении резервирования "верхний резервирует нижнего". Поток отказов ячеек предполагаем пуассоновским. Вероятность отказа будем определять как функцию относительного времени $\tau = \frac{t}{t_{отк}}$, где $t_{отк} = \frac{1}{\lambda}$ - время наработки на отказ (λ - интенсивность отказов). Без резервирования отказ одной ячейки приведет к невозможности выполнения системой части функций, поэтому будем считать это отказом системы. Вероятность такого отказа [4]:

$$p(\tau) = \sum_{i=1}^{N} p_i(\tau) = \sum_{i=1}^{N} \frac{\tau^i}{i!} 1^{-\tau} = 1^{-\tau} \sum_{i=1}^{N} \frac{\tau^i}{i!},$$

где N - число ячеек системы, $p_i(\tau)$ - вероятность отказа i ячеек за время τ.

Для определения вероятности отказа системы резервирования предположим, что система состоит из набора "кустов" ячеек. Каждый "куст" включает в себя одну ячейку на вершине и α ячеек в основании, где α - средний коэффициент ветвления. Отказ двух ячеек любого "куста" приводит к его отказу и отказу всей системы. Определим вероятность этого события.

Предположим, что за время τ отказало i ячеек. Вероятность того, что отказ произошел в данном "кусте", равна $\xi = \frac{\alpha+1}{N}$; вероятность того, что из i отказов j попало в данный "куст", подчиняется биномиальному закону распределения $p'_{ij} = C_i^j \xi^j (1-\xi)^{i-j}$.

Вероятность того, что такой отказ произошел в каком-нибудь из Q кустов, $p_{ij} = C_i^j Q\, \xi^j (1-\xi)^{i-j}$, где Q - число "кустов" системы. Общая вероятность отказа системы при резервировании $p_{отк}(\tau)$ за время τ равно взвешенной по i сумме вероятностей p_{ij}

$$p_{отк}(\tau) = \sum_{i=2}^{Q} p_i(\tau) \sum_{j=2}^{\alpha+1} p_{ij} + \sum_{i=Q+1}^{N} p_i(\tau) =$$

$$= 1^{-\tau}\left[Q \sum_{i=2} \frac{\tau^i}{i!} \sum_{j=2}^{\alpha+1} C_i^j \xi^j (1-\xi)^{i-j} + \sum_{i=Q+1}^{N} \frac{\tau^i}{i!}\right]. \quad (1)$$

Второй член выражения соответствует вероятности отказа более чем $Q+1$ ячеек, что приводит к безусловному отказу системы. Число кустов Q может быть определено из следующих соображений. Оно меньше числа N на число ячеек самого нижнего уровня системы. Поэтому

$$N = \sum_{i=1}^{m-1} \alpha^i = \frac{\alpha^m - 1}{\alpha - 1}, \qquad Q = \sum_{i=1}^{m-2} \alpha^i = \frac{\alpha^n - 1}{\alpha - 1},$$

где m - число уровней системы. Исключая m из этих уравнений, получаем:

$$Q = N - \alpha^{\frac{\ln[N(\alpha-1)+1]}{\ln \alpha}} = \frac{N-1}{\alpha}.$$

Для расчетов по формуле (I) значение Q округлено по формуле $Q = Ent\left(\frac{N-0.5}{\alpha}\right)$.

Расчеты производились для следующих значений аргументов: N = I0,20,30,40,50; α = 2,3,4,5; τ = 0,2; 0,4; 0,6; 0,8; I,0; I,4; I,8; 2,2; 2,6; 3,0; 4,0; 5,0; 6,0; 7,0; 8,0.

Некоторые результаты расчетов сведены в табл.2.

Таблица 2

Число ячеек	Средний коэффициент ветвления	τ - относительное время					
		0,2	0,4	0,8	I,4	3,0	8,0
I0	2	0,0078	0,027	0,084	0,I72	0,33	0,66
	3	0,0II	0,039	0,I2	0,255	0,533	0,8I
	4	0,0I3	0,046	0,I45	0,3I9	0,688	0,92
	5	0,0I3	0,0465	0,I50	0,34	0,737	0,96
20	2	0,0039	0,0I37	0,04I9	0,085	0,I32	0,235
	3	0,0049	0,0I7I	0,0522	0,I06	0,I76	0,59
	4	0,0055	0,0I9I	0,0585	0,I22	0,262	0,84
	5	0,0063	0,022	0,0684	0,I49	0,379	0,92
40	2	0,00I9	0,0068	0,0209	0,043	0,068	0,I2
	3	0,0023	0,0079	0,024	0,049	0.078	0,I32
	4	0,0027	0,0095	0,029	0,059	0,093	0,I49
	5	0,003I	0,0II	0,033	0,068	0,II	0,443
Без резервирования		0,I8	0,37	0,59	0,72	0,95	0,99

В последней строке табл.2 приведены значения вероятности отказа без резервирования, эта вероятность зависит только от τ. Видно, что выигрыш в надежности для $\tau = I$ весьма значителен и тем больше, чем больше Q. Даже для минимальных Q (порядка 3) вероятность отказа уменьшается в 5-6 раз, а для $Q = 20$ уменьшается больше чем на порядок. Вообще говоря, общее количество отказавших ячеек при сохранении работоспособности может достигать $\frac{N-1}{\alpha}$, но при этом отказавшие ячеки должны быть определенным образом "расставлены".

Структура ячейки (рис.3) является достаточно традиционной.

Особенностью является наличие сепаратора интерфейса (переключателя шины), который по команде вышестоящего уровня может объединять интерфейсные шины для получения доступа к ОЗУ нижестоящего уровня. Сепаратор служит как для обмена информацией, так и для резервирования. При этом для исключения неоднозначностей (при передачах управления и т.д.) адресные ресурсы должны распределять-

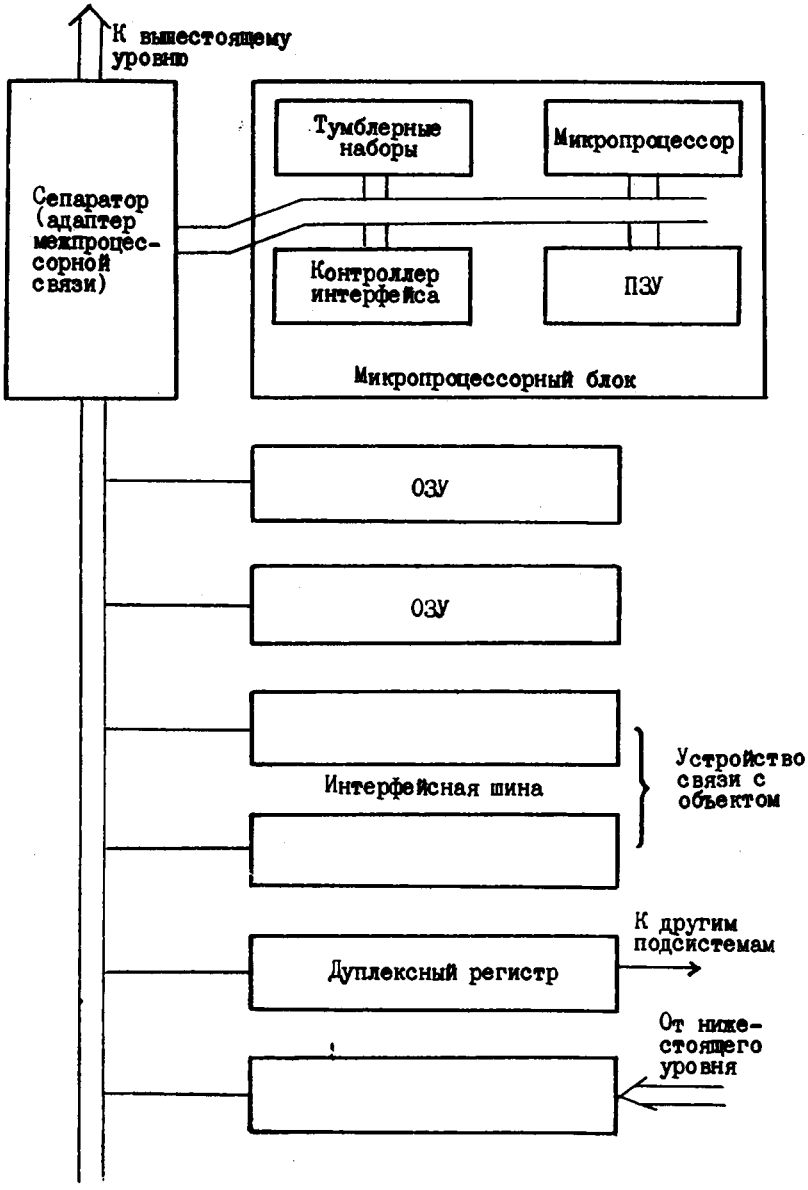


Рис.3. Структура ячейки.

ся между двумя соседними уровнями. Возможное распределение адре - сов должно укладываться в систему неравенств:

$$a_1 + a_2 \le n,$$
$$a_2 + a_3 \le n,$$
$$a_{i-1}+a_i \le n,$$
$$a_i+a_{i+1} \le n,$$

где a_i - размер адресного поля i-го уровня, n - размер адресуемого пространства используемого микропроцессора, т.е. адресный ресурс.

Рассмотрим возможный вариант заполнения адресного поля ячейки (подсистемы). Все программное обеспечение может быть отнесено к одной из двух групп в зависимости от того, изменяется ли оно в процессе эксперимента. Такое разделение необходимо для определения типа памяти (ОЗУ,ППЗУ,ПЗУ) и способа резервирования. Постоянные программы, образующие "начальный интеллект" ячейки,включают в себя программы,обеспечивающие функционирование ячейки, ее связь с другими ячейками и т.д. Эти программы размещаются в ПЗУ. Полупостоянные программы отражают общение ячейки с объектом; это даже не программы, а таблицы, которые могут храниться как в ОЗУ, так и в ППЗУ с электрической перезаписью. Эта часть программного обес - печения заполняется при генерации системы.

Изменяющаяся часть программного обеспечения должна размещаться в ОЗУ. Это, в первую очередь, программы эксперимента, в том числе обработчики прерываний. Структура системы позволяет изменять эти программы по желанию экспериментаторов в процессе эксперимента. Наконец, наиболее изменчивая часть информации - те таблицы и данные, которые отражают прохождение эксперимента.

Необходимо обеспечить восстановление любой информации в случае выхода из строя модуля ПЗУ, ОЗУ или ППЗУ. В случае отказа ПЗУ управление передается вышестоящей ячейке, имеющей такой же набор управляющих и других системных программ. Таблицы подсистемы и пользовательские программы хранятся в архиве эксперимента, на внешнем носителе пульта экспериментатора. При отказе модуля,содержащего эту информацию, она перезагружается в резервный модуль ОЗУ, имеющийся в каждой паре уровней. Данные, подвергающиеся непрерывным изменениям, должны дублироваться внутри ОЗУ каждой ячейки, в различных ее модулях.

Распределение памяти в паре соседних уровней сводится, таким образом, к определению границы между полями уровней. Так как большинство микропроцессоров имеют I6 адресных шин, то n = 64 Кбайт, из которых под постоянные программы необходимо выделить от 4 до 8 Кбайт. Адреса регистров устройств могут занимать область до 8 тысяч адресов; таким образом, между уровнями должны быть распределены 48 Кбайт. Для задач взаимодействия с объектом этого, как правило, достаточно.

Программное обеспечение информационно-управляющей системы состоит из двух основных частей:

- базового постоянного обеспечения ячейки;
- системных средств доступа.

Первое обеспечивает начальный "интеллект" ячейки и включает в себя супервизор программ (обеспечивающий "круговорот" программ), программы обмена с верхним и нижним уровнями, программу загрузки программ и данных, программы резервирования и супервизор событий. Общий объем этих программ от 4 до 8 Кбайт, они находятся в постоянном запоминающем устройстве ячейки.

3. Системные средства доступа обеспечивают решение двух задач:

- автоматизируют программирование взаимодействия системы с объектом;
- позволяют увеличить тезаурус системы относительно себя и объекта.

Программы, реализующие эти средства, размещаются, в основном, в пульте оператора-экспериментатора.

Первая задача решается введением языка реального времени [5], обеспечивающего взаимодействие оператора с объектом на уровне символьно-логических обозначений. Транслятор должен располагать информацией об отображении логических имен точек ввода и вывода информации на физическую реализацию связи с объектом. Это - нулевой уровень тезауруса, обеспечивающий ассоциативность программ. Первый уровень - сведения о том, какие ограничения имеются на изменения физических величин во всех точках соприкосновения объекта с системой. Два этих уровня могут быть обеспечены использованием бланк-языков и реализоваться непосредственно в ячейках системы. Второй уровень связан с классификацией образов входных воздействий и выходных величин в соответствии с определенными заранее множества - ми. Эта задача связана с бо́льшим объемом вычислений, чем предыду-

щие, хотя решается известными методами конечномерных евклидовых пространств [6]. В терминах формальных грамматик эта задача может быть сформулирована как задача лексического анализа выражений на входном и выходном языках объекта.

Третий уровень тезауруса включает в себя контроль динамических параметров входных и выходных воздействий объектов, т.е. классификацию образов траекторий в пространствах входных воздействий и выходных величин. Это - задача синтаксического анализа, и для ее решения могут применяться известные приемы трансляции, требующие, однако, еще больших затрат памяти и машинного времени.

Четвертый уровень связан с идентификацией объекта и предполагает контроль соответствия входных траекторий выходным. Функционирование широкого класса стационарных динамических объектов может быть описано с помощью автоматных грамматик, поэтому задача такого контроля может быть сформулирована как задача грамматического вывода. Следует отметить слабое развитие программных средств для классификационных задач такого рода [7], по-видимому, в этом направлении необходимы серьезные исследования.

Введение описанных уровней позволяет существенно повысить надежность программ реального времени за счет замедления выполнения программ. Многопроцессорность системы в некоторой степени компенсирует этот недостаток, однако эффективное управление объектами, требующими возобновления управляющей информации с интервалом единиц миллисекунд, с помощью "обычных" микропроцессорных средств с быстродействием 100-200 тыс.опер./сек, по-видимому, невозможно. Можно предположить, что реализация третьего и четвертого уровня тезауруса в реальном времени потребует привлечения мощных вычислительных средств типа перестраиваемых вычислительных сред [8].

Выводы

1. Описанная информационно-управляющая система обладает большим количеством степеней свободы, которые позволяют легко настраивать ее на любую конфигурацию объекта.

2. Структура системы - дерево с возможностью расширения до ориентированного графа. Такая структура выгодна при большом количестве независимых "замкнутых" контуров переработки информации, не замыкающихся через экспериментатора.

3. При применении резервирования микропроцессорного блока и модуля ОЗУ система оказывается достаточно надежной. Без увеличе -

ния числа микропроцессорных блоков время наработки на отказ уменьшается в несколько раз. Живучесть сохраняется при одновременном отказе до $\frac{1}{4}-\frac{1}{3}$ общего количества ячеек.

4. Предлагаемая система может быть реализована на современных средствах микропроцессорной техники. Размеры адресного поля каждой подсистемы в среднем 24 Кбайт.

5. Благодаря быстрой настройке и перестройке такая структура может применяться не только для автоматизации экспериментов, но и в АСУ ТП. После проведения отладочных работ программы могут быть записаны в ППЗУ, а пульт экспериментатора заменен набором функциональных клавиш и табло коллективного пользования.

6. Рассредоточенность вычислительных средств позволяет повысить надежность программ путем введения контролирующих процедур, основанных на использовании априорной информации об объекте (стенде) как о динамической системе.

Литература

1. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности.-Новосибирск:Наука,1966.- 308 с.

2. МЕСАРОВИЧ М., МАКО Д., ТАКАХАРА И. Теория иерархических многоуровневых систем / Под ред. И.Ф. Шаянова. - М.: Мир, 1973. - 344 с.

3. МАРТИН Дж. Организация баз данных в вычислительных системах / Под ред.А.Л.Щерса.- М.:Мир, 1978. - 616 с.

4. КОРН Г., КОРН Т. Справочник по математике для научных работников и инженеров / Под ред.И.Г.Арамановича. - М.: Наука,1977. - 830 с.

5. БЕРЕЗИН Б.А., ЦЫПИНСКИЙ Б.Г. Подход к программированию систем реального времени на базе микро-ЭВМ. - Управляющие системы и машины, 1981, № 5, с.76-82.

6. КАЛМАН Р., ФАЛБ П., АРБИБ М. Очерки по математической теории систем / Под ред. Я.З.Цыпкина.- М.:Мир, 1971.- 400 с.

7. ХАНТ Э. Искусственный интеллект / Под ред. В.Л.Стефанюка. - М.:Мир,1978. - 558 с.

8. ПРАНГИШВИЛИ И.В., СТЕЦЮРА Г.Г. Микропроцессорные системы. - М.: Наука, 1980. - 237 с.

Поступила в ред.-изд.отд.
22 марта 1982 года