УДК 004.2 DOI: 10.17587/it.24.763-771

А. Б. Барский, д-р техн. наук, проф., e-mail: arkbarsk@mail.ru,
Д. И. Мельник, канд. техн. наук, ст. науч. сотр., e-mail: mdi_dim@mail.ru,
НИИЦ (г. Москва) ЦНИИ ВВКО Минобороны РФ

Сетевая технология проектирования асинхронной вычислительной системы на общем решающем поле с самоуправляемой параллельной обработкой данных

Исследуется возможность построения макета управляющей двухпроцессорной вычислительной системы, обладающей общим решающим полем, с асинхронной обработкой данных по принципу data flow. Макет выполняется на базе мультимикропроцессорной системы "Сивуч-1", объединяющей 36 микропроцессоров "Эльбрус". Цель макетирования — создание инструментального средства для экспериментальных исследований в области достижения высокой производительности вычислительных средств и специального программного обеспечения, а также высокоустойчивых и безопасных вычислений.

Ключевые слова: *сетевая технология, макетирование, data flow, процессор, вычислитель, процессор памяти*

Введение

Реализация проектов гонки вооружений конца 70-х — начала 80-х гг. XX века стимулировала интенсивный поиск принципиально новых решений на пути создания высокопроизводительных вычислительных средств[1]. В то время общее признание получил так называемый "японский вызов" (опубликован на русском языке в 1980 г.). В рамках этого вызова были сформулированы два пути решения проблемы:

1. Объединение в одной установке десятков тысяч микропроцессоров, где бы совместное решение задач регулировалось по принципу *data flow*.

2. Применение нейросетевых технологий для решения задач высокой сложности, трудноформализуемых задач и задач управления в реальном времени методом ассоциативных вычислений.

Первые научно-исследовательские разработки [1] в области "потоковых машин", высокая эффективность которых подкреплена результатами моделирования [2, 3], не вышли на экспериментальный уровень.

В настоящее время появились вычислительные средства, представляющие собой комплексы микропроцессоров "Эльбрус", объединенные на базе сетевой технологии *Ethernet*. К ним относится вычислительный комплекс (ВК) "Сивуч-1", который можно рассматривать как мини-сеть. Такая мини-сеть является не только превосходным средством моделирования распределенных систем и процессов, но позволяет на новом уровне рассмотреть возможность макетирования перспективных вычислительных систем (ВС) с учетом их функционирования в сложных управляющих системах.

1. Обоснование разработки макета

В предложенном ранее проекте [1—3] была реализована концепция мультимикропроцессорной системы на общем решающем поле [4]. Самоуправляемая параллельная обработка осуществлялась по "классической" схеме *data flow*. По этой схеме каждая команда не использует адреса операндов, а содержит места для размещения этих операндов, куда они поступают в результате вычислений или из памяти.

[1] Производительность вычислительного средства устанавливается на основе выполнения "машинных" или эквивалентных алгоритмических операций в секунду, а также по временным характеристикам контрольных (функциональных) или тестовых задач.

Наличие всех операндов в команде позволяет приступить к ее выполнению. В динамике вычислений готовыми к выполнению одновременно могут быть несколько команд. Результат их выполнения поступает либо в память (в текст заявки), либо в текст других, заранее распределенных команд. Вычислительный процесс оказывается связанным так, что естественным образом реализуются как параллельные, так и последовательные действия алгоритма.

Однако на этом применение "классической" идеи было закончено, так как ни одна известная попытка ее эффективной реализации не достигла успеха. Помехой тому служило резкое повышение требований к системе оперативного обмена информацией большим числом узлов внутри ВК, исключившее возможность практического использования такой привлекательной (с точки зрения программистов-математиков) идеи.

Поэтому авторами проекта было принято компромиссное решение (рис. 1), суть которого состоит в следующем.

Все микропроцессоры (вычислители), образующие решающее поле ВС (общее арифметическо-логическое устройство, АЛУ), обладают буферами, в которых динамически, в процессе выполнения программы процессором (процессорами — в многопроцессорной ВС), практически выполняющим функции устройства управления (УУ), формируются множества команд вида, свойственного *data flow*. На рис. 1 показан общий вид команды, где θ — код операции, r_1, ..., r_4 — операнды, занявшие свои места, R — временно хранящийся результат, A_R — адрес, по которому результат должен быть отправлен.

Одним из вычислителей является *процессор памяти* (ПП). Как и вычислитель, он выполняет заявки асинхронно по мере поступления необходимой информации, но соблюдая частичную упорядоченность операций над памятью, обусловленную программой. В ходе анализа каждой команды ПП формирует заявки к оперативной *памяти данных* (ОПД), располагая адресом регистра буфера вычислителя, куда он должен послать результат считывания. Для выполнения необходимой записи в память вычислитель располагает адресом заявки на запись в буфере ПП (в *очереди заявок* (ОЗП)), куда он пошлет нужный результат. Затем эта заявка может быть рассмотрена для выполнения.

Таким образом, команды оказываются, во-первых, заранее, до установления их готовности к выполнению, распределенными между вычислителями, а во-вторых, динамически связанными между собой промежуточными результатами счета и с памятью данных в соответствии с реализуемым алгоритмом. Процесс коммутации вычислителей и процесс выполнения команд являются асинхронными. Процессоры наполняют буферы вычислителей и связывают их между собой, а также с памятью данных, а вычислители выполняют готовые команды в своем буфере и посылают результаты по сформированным адресам.

Программа такой ВС (программа коммутации) имеет традиционный вид и не предъявляет каких-либо особых требований к языкам программирования. Следует лишь отметить наличие в оттранслированной программе данных специального типа "*вычислитель*". Виртуальным адресом вычислителя удобно именовать результат промежуточных вычислений, подобно условным адресам "рабочих ячеек" при программировании ранних ЭВМ. Присутствующие в программе виртуальные адреса вычислителей образуют условную память, динамически распределяемую адресным генератором в процессе вычислений. Данные в этой памяти обладают короткой жизнью: от объявления до первого использования. После этого виртуальное имя вычислителя может быть использовано вновь.

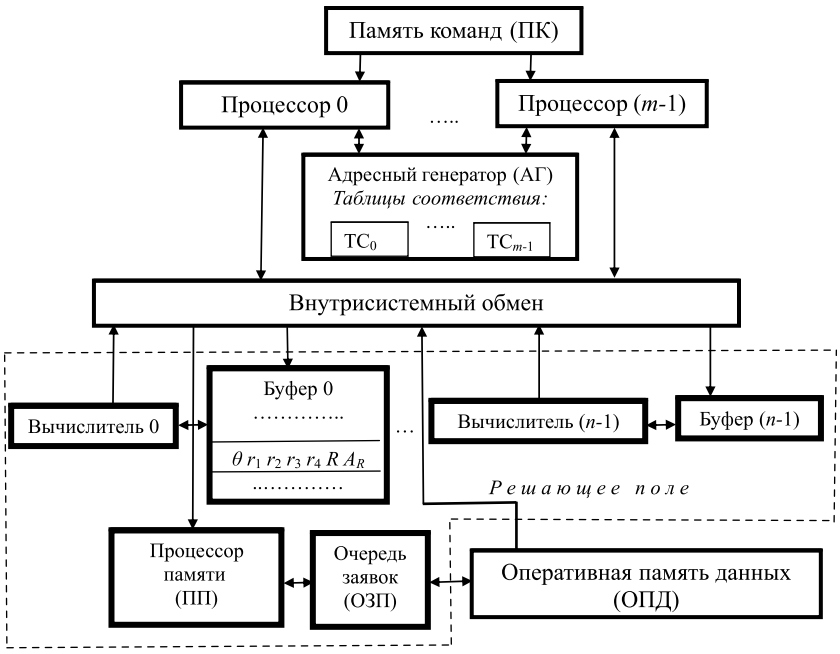

Рис. 1. Схема макетируемой вычислительной системы

Говоря об эффективности трансляции, следует отметить важность промежуточного представления алгоритма в бесскобочной (инверсной) записи ПОЛИЗ (предложил польский математик Ян Лукашевич в 1929 г.), что характерно для распространенной технологии трансляции. Тогда при переходе к физическим ресурсам следующие подряд "параллельные" команды оказываются распределенными на "параллельные" вычислители, если их достаточно.

Для понимания принципиальной схемы вычислений кратко представим рассмотренный в работе [1] упрощенный пример программирования и даже трансляции для однопроцессорной ВС рассмотренного типа.

Пусть команда программы коммутации имеет традиционную трехадресную структуру $\theta\ A_1\ A_2\ A_3$, где A_1 и A_2 – адреса операндов (ОПД или регистра буфера вычислителя), A_3 – адрес регистра буфера того вычислителя, который будет выполнять инструкцию, сформированную на основе этой команды. Не рассматривая виртуализацию ресурса, адреса вычислителей будем задавать парой {номер вычислителя, номер регистра его буфера}.

Пусть при $n = 4$ следует написать программу счета значения выражения

$$A: = ((a + b)(c + d)(e + f + g) - hij):(k(l - \sqrt{m})).$$

Как известно, в результате преобразования в ПОЛИЗ, на основе этой записи легко получается программа в безадресной системе команд, соответствующая выполнению операций на стеке:

$$ab + cd + \times ef + g + \times hi\times j\times - klm\sqrt{}-\times:\underline{Зап}A,$$

где $\underline{Зап}A$ — команда "запись по адресу A".

При последовательном, послойном использовании регистров буферов вычислителей (в примере нумерация начинается с единицы) программа (коммутации) примет вид, показанный на рис. 2.

На рис. 3 представлена схема динамического формирования и коммутации инструкций в регистрах I_{ij} ($i, j = 1, ..., 4$) буферов вычислителей и в очереди заявок к памяти. Для упрощения примера рассмотрены укороченные регистры, вмещающие только два операнда. В клетках попарно объединены заявки, формируемые при анализе команд программы. Знак *Сч* означает считывание, знак *Зап* означает запись по адресу памяти. Динамика выполнения команд и заявок к памяти не отслеживается.

N_k	КОП	A_1	A_2	A_3
1	+	***a***	***b***	**(1, 1)**
2	+	***c***	***d***	**(2, 1)**
3	+	***e***	***f***	**(3, 1)**
4	×	***h***	***i***	**(4, 1)**
5	√	***m***		**(1, 2)**
6	×	**(1, 1)**	**(2, 1)**	**(2, 2)**
7	+	**(3, 1)**	***g***	**(3, 2)**
8	×	**(4, 1)**	***j***	**(4, 2)**
9	–	***l***	**(1, 2)**	**(1, 3)**
10	×	**(2, 2)**	**(3, 2)**	**(2, 3)**
11	×	***k***	**(1, 3)**	**(3, 3)**
12	–	**(2, 3)**	**(4, 2)**	**(4, 3)**
13	:	**(4, 3)**	**(3, 3)**	***A***

Рис. 2. Программа коммутации

Запрограммируем счет выражения

$$A := if\ (a + b) > c\ then\ if\ (c - d) > e\ then\ X\ else\ (B - D) else\ Q.$$

Пусть команда *УСЛ* $A_1\ A_2\ A_3\ A_4\ A_5$ записывается в двух словах и интерпретируется как $A_3 := if\ (A_1) > (A_2)\ then\ (A_4)\ else\ (A_5)$.

Сформируем три команды для коммутации счета арифметических операторов, составляющих это выражение (рис. 4). Затем последовательно используем две команды *УСЛ* для коммутации внутренней и внешней конструкций типа "*if — then — else*".

Как видно, ни одной команды условного перехода, задерживающей конвейер выполнения команд, применить не пришлось.

Для составления программы в математических (виртуальных) адресах вычислителей предположим, что мы располагаем единственным условным вычислителем с буфером, объем которого определен специальной областью адресного пространства. Средствами виртуализации вычислительного ресурса в ВС, содержащей m процессоров коммутации, являются адресный генератор (АГ) и таблицы соответствия $ТС_k$, $k = 0, ..., m - 1$.

Если любой адрес A_μ, $\mu = 1, 2, 3$, команды программы, выполняемой на k-м процессоре, является математическим адресом вычислителя, выполняется обращение к $ТС_k$, состоящей

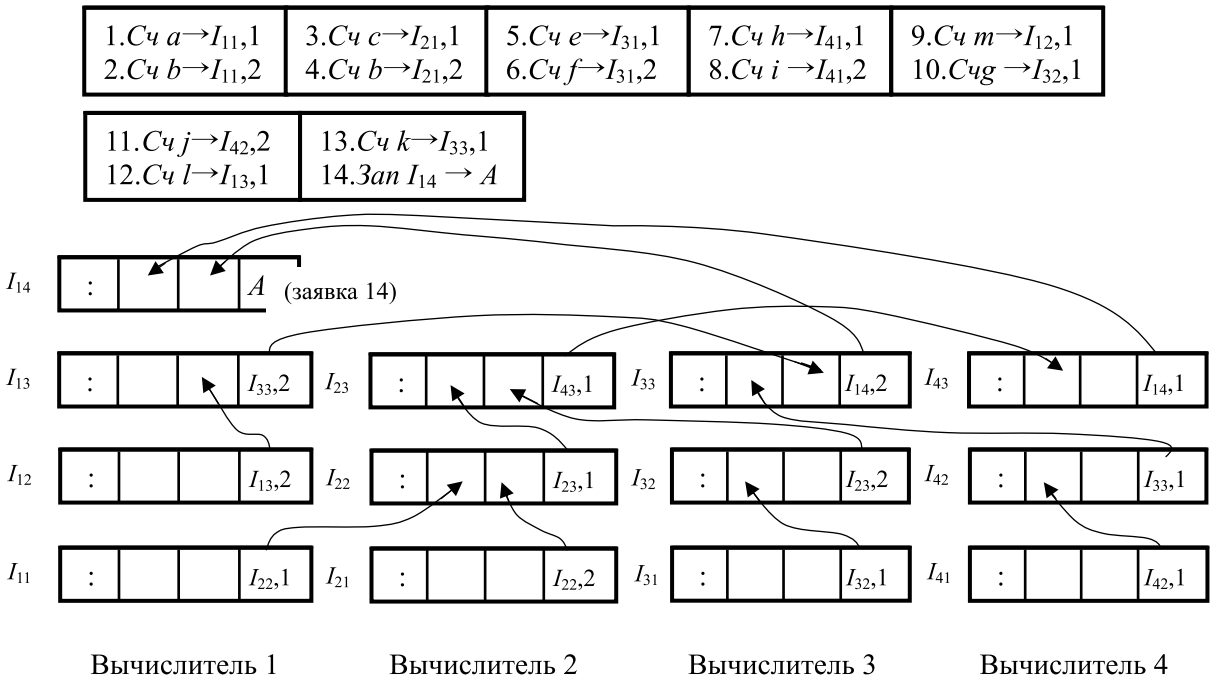

Рис. 3. Схема коммутации вычислителей

1.	+	*a*	*b*	(1, 1)
2.	-	*c*	*d*	(2, 1)
3.	-	*B*	*D*	(3, 1)
4.	***УСЛ***	(2, 1)	*e*	(4, 1)
		X	(3, 1)	
5.	***УСЛ***	(1, 1)	*c*	*A*
		(4. 1)	***Q***	

Рис. 4. Ветвление без условных переходов

из строк соответствия вида $v_i \rightarrow (i_j, s_j)$, где v_i – математический адрес некоторого вычислителя; (i_j, s_j) – соответствующий ему физический адрес. В случае совпадения $A_\mu = v_i$ фиксируется найденный физический адрес, а использованная строка исключается из $ТС_k$. Необходимость исключения строки следует из того, что каждый результат вычислений используется лишь один раз. При безуспешном обращении к $ТС_k$ АГ выдает очередной физический адрес вычислителя в порядке последовательной загрузки вычислителей решающего поля и при наличии свободных регистров в их буферах. Одновременно для данного математического адреса и найденного физического в первом свободном регистре $ТС_k$ формируется новая строка соответствия.

Если по третьему адресу команды не указан математический адрес вычислителя (например, указан адрес ОПД), то АГ в порядке последовательной загрузки вычислителей формирует физический адрес вычислителя-исполнителя данной команды. По этому адресу записывается инструкция для выполнения операции, указанной в команде.

В рассмотренной в работах [1—3] модели многопроцессорной ВС (см. рис. 1) предполагалось, что АГ является автономным быстродействующим устройством, которое в соответствии с последовательной загрузкой вычислителей выдает всем процессорам одновременно адреса вычислителей для их использования при обработке адресов команд. Для ускорения его работы предлагалось использовать метод ситуационного управления, реализованный на ассоциативной памяти. Однако в макете предложен другой, значительно более простой способ виртуализации адресов вычислителей.

Упорядочение во времени выполнения заявок к ОПД, в которых указан один и тот же адрес, обеспечивается тем, что при выборе для исполнения адрес ячейки ОПД, указанный в каждой заявке, сравнивается с адресами ячеек ОПД, указанными в еще не исполненных, т. е. находящихся в ОПД, заявках, пришедших

ранее. Заявка может быть выполнена лишь в том случае, если выполнены все заявки по указанному в ней адресу, поступившие ранее.

2. Требования к макету

В основу разработки макета положены следующие требования:

1. Вычислительная система управления в реальном времени должна набираться из стандартных модулей в соответствии с функциональным назначением. Программы ВС должны быть инвариантны относительно размерности данных и относительно числа вычислителей.

2. Накладные расходы на управление параллельной работой большого числа модулей ВС должны быть минимальными.

3. Формируемые вычислительные установки должны соответствовать требованиям управления надежностью (избыточной комплектацией, ремонтопригодностью, составом ЗИП).

4. Разработка управляющей ВС диктует возврат к жесткому планированию использования оперативной памяти (ОП). Это связано с современными технологическими возможностями создания больших объемов прямоадресуемой, достаточно быстродействующей, структурированной ОП. Так, если в МВК "Эльбрус-2" объем разделяемой ОП составлял 144 Мбайт, то в ВК "Эльбрус 401-*PC*" оперативная память составляет 24 Гбайт. Отказ от трудоемких механизмов ведения (с помощью операционной системы (ОС)) виртуальной памяти обеспечивает весьма ощутимый резерв производительности для режима реального времени.

5. ОС не должна участвовать в вычислениях.

6. Система прерывания должна обеспечивать только временно́й режим и обслуживать только чрезвычайные ситуации.

7. Стек процедур, в прошлом вызывающий нарекание разработчиков сложных систем, должен быть реализован только в том случае, если он не поддержан ОС в режиме управления в реальном времени. (Замечательная команда "Передача управления с возвратом", например в БЭСМ-6, хорошо поддерживала высокую культуру программиста при структуризации программы для выделения "стандартных подпрограмм" – неоднократно используемых участков, которые можно интерпретировать процедурами.)

8. Необходимо снизить влияние условных переходов, задерживающих вычислительный процесс, вводом команды "*if-then-else*", допускающей любую структуру вложений условного арифметического оператора присваивания.

9. Передача управления по адресу, являющемуся переменной с индексом, позволяет быстро переключать каналы обслуживания, если индекс канала совпадает с параметром цикла.

3. Инструментальное средство макетирования и структура макета

Возврат к идее построения ВС на указанных принципах сегодня обусловлен наличием инструментальных средств, позволяющих с уровня моделирования [3] перейти на уровень макетирования, т. е. создания "живой" архитектуры ВС, позволяющей реализовать рабочий режим, поиск оптимальных архитектурных и схемотехнических решений, внедрять предложения по организации вычислительного процесса, построения операционной системы, по обеспечению надежности, внедрять результаты других исследований.

Очевидно, на макете нельзя воспроизвести ожидаемые достоинства целевой ВС в части производительности, однако *коэффициент полезной загрузки оборудования* может быть определен с высокой достоверностью.

Предполагается в качестве инструментального средства макетирования взять многопроцессорный ВК "Сивуч-1" с распределенной памятью [5].

Данный ВК предназначен для решения задач многоканальной цифровой обработки больших объемов информации в режиме реального времени, для функционального и технического управления в области радиолокации, управления оптико-электронными и радиотехническими средствами, для реализации универсальных алгоритмов обработки информации на пунктах управления сложными системами.

Тридцать шесть процессоров МП1С1/V, объединенных в *три устройства вычислителя* УВ/С в его составе, связаны между собой на базе сетевой технологии *Ethernet* с помощью коммутатора *GbEthernet KGE*. Они условно образуют фрагмент локальной вычислительной сети. Таким образом, ВК "Сивуч-1" является хорошим инструментальным средством нового направления – сетевого моделирования.

На этапе макетирования двухпроцессорной асинхронной системы представляется достаточным использование одного устройства УВ/С, содержащего 12 процессорных модулей МП1С1/V, объединенных, в свою очередь, в три ВС МП4С1/V *SYSTEM* 0, *SYSTEM* 1,

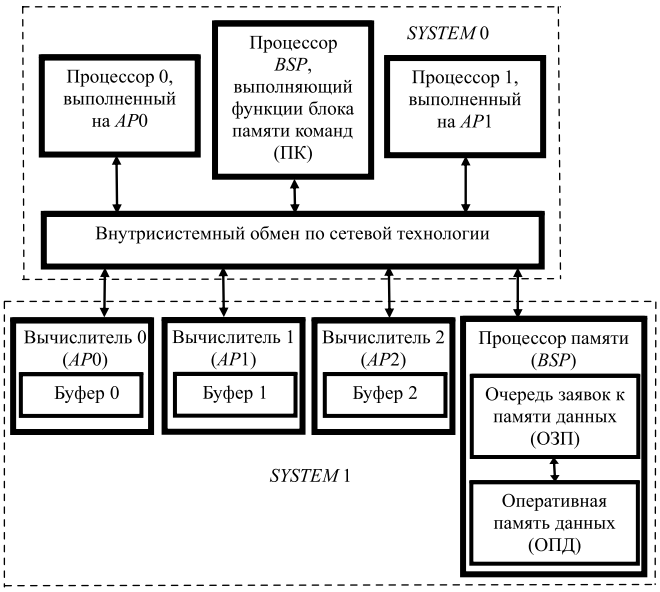

Рис. 5. Распределение функций устройств макетируемой ВС между процессорами ВК "Сивуч-1"

SYSTEM 2. Один модуль МП1С1/V ВС является *главным* (*Bootstrap, BSP*). Он выполняет начальную загрузку. Каждый из трех *дополнительных модулей* называется *Application* (*AP*0, *AP*1, *AP*2).

На рис. 5 представлена схема распределения функций макета среди вычислительных модулей одного УВ/С ВК "Сивуч-1". Для первой очереди макетирования достаточно двух ВС — *SYSTEM* 0 и *SYSTEM* 1.

Как сказано выше, найдено решение, исключающее необходимость построения АГ в качестве отдельного устройства.

4. Структура и система команд

Команды макетируемой асинхронной ВС бывают нуль-, одно-, двух-, трех- и пятиадресными. Трехадресная команда образует командное слово основного формата

$$\theta\ J_1 A_1\ J_2 A_2\ J_3 A_3,$$

где θ — код операции, J_j, $j = 1, 2, 3$, — адреса регистров-модификаторов, A_j — смещения.

Для каждой команды формируются математические адреса

$$A'_j = (J_j) + A_j,\ j = 1, 2, 3.$$

Пятиадресная команда занимает два командных слова

$$\theta\ J_1\ A_1\ J_2\ A_2\ J_3 A_3$$
$$0\ J_4\ A_4\ J_5\ A_5\ 0\ 0.$$

Адреса A'_1, A'_2, A'_4, A'_5 определяют операнды. В арифметических, логических и операциях отношения эти адреса могут быть математическими (виртуальными) адресами вычислителей или ОПД, адреса A'_2 и A'_5, кроме того, могут быть кодами целых чисел, непосредственно участвующих в операции, а также адресами модификаторов. Адрес A'_3 определяет результат и может быть адресом ПК, модификатора, ОПД, вычислителя, на котором выполняется данная операция и по адресу которого, следовательно, присваивается значение результата операции.

При макетировании предполагается, что указанный выше тип адресной информации определяется по тегам. Например, два старших дополнительных разряда могут определять теги: 00 — число или адрес ПК, 01 — адрес модификатора, 10 — адрес ОПД, 11 — адрес вычислителя.

В работах [2, 3] подробно обсуждается весьма развитая система команд арифметических и логических операций, операций отношения, условные арифметические операции, операции пересылки и преобразования кодов, операции индексации, групповые, векторные операции и операции цикла.

Там же подробно обсуждаются команды передачи управления. Хотя механизм выполнения вложенных процедур на стеке подвергается критике в применении к управляющим вычислительным средствам, целесообразно для универсальных применений воспользоваться возможностями, уже заложенными в микропроцессорах "Эльбрус" ВК "Сивуч-1". Поэтому предлагаются такие команды, как переход с возвратом внутри процедуры и переход на следующий лексикографический уровень с возвратом. Команда условного перехода ставится в зависимость от сигнала о выработке условия. До появления условия перехода могут выполняться другие команды.

Впервые в соответствии с концепцией системы управления как многоканальной системы массового обслуживания предлагается использовать *команду переключения каналов.* По этой команде запоминается текущее значение регистра команд и счетчика команд для программы обрабатываемого канала и вводятся или восстанавливаются соответствующие значения для программы обработки канала следующего номера (по mod <*число каналов процессора*>). С каналами связаны базовые регистры, указывающие на их данные. Выполнение этой команды в цикле позволяет внутри одного

такта управления обработать все объекты, закрепленные за каналами в соответствии со стадией их обслуживания. Один канал головного процессора П0 может закрепляться за супервизором, отслеживающим обстановку и принимающим решения по дальнейшему ходу управления системой: о переходе на следующие стадии обслуживания объектов, о включении новых объектов, о реконфигурации системы и т. д. По всем каналам процессора П1 может проводиться только обслуживание.

С учетом возможностей микропрограммирования следует сделать вывод о целесообразности такого развития системы команд, чтобы обусловленные ими работы были достаточно объемны, существенно загружали вычислители, снижали относительные затраты на управление распараллеливанием, по смыслу приближались к реальным процедурам [3]. Ряд команд должен соответствовать выполнению многих элементарных функций. Специальные применения ВС диктуют и специализацию развиваемой системы команд.

5. Алгоритмы работы устройств макета

1. Обработка команд процессором. Три уровня обработки команд процессором подробно рассмотрены в работе [1]. Однако в связи с отсутствием АГ в макете предлагается другая схема формирования физических адресов вычислителей, реализуемая совместно процессором и выделенным им вычислителем. Вычислитель сообщает процессору адрес регистра, в котором будет формироваться инструкция вычислителю по анализируемой процессором команде. По этой информации в таблице соответствия процессора $ТС_k$, $k = 0, 1$, формируется строка вида "$v \rightarrow$ (номер вычислителя, адрес его буфера)", где v — встретившийся в программе виртуальный адрес вычислителя, выполняющего данную команду.

Рассмотрим эту схему подробнее.

При анализе очередной команды k-м процессором первый раз выбирается вычислитель $j = k$. Затем проводится переадресация для выбора вычислителя при анализе следующей команды: $j := (k + 2)_{mod2}$. Таким образом осуществляется распределение вычислителей, подобно тому, как распределяются данные между исполнителями по *SPMD*-технологии.

Вычислитель воспринимает свое назначение как заявку на предоставление свободного регистра своего буфера для организации в нем инструкции вычислителю, соответствующей данной команде процессора. Для этого все адреса регистров буфера динамически образуют два списка: *список используемых адресов* и *список освободившихся адресов*. Когда список используемых адресов иссякает, список освободившихся адресов становится списком используемых адресов, а опустевший список используемых адресов становится списком освободившихся адресов.

На рис. 6, *а* для буфера, состоящего из шести регистров, показано состояние указанных списков в момент начала вычислений. На рис. 6, *б* показано возможное состояние списков в некоторый момент вычислений с учетом разного времени выполнения инструкций.

2. Обработка инструкций вычислителями. Вычислитель занимает выделенный регистр пришедшим от процессора кодом операции и передает сообщение этому процессору для формирования строки соответствия в его ТС, в каком регистре располагается новая инструкция. Поскольку каждая инструкция готова к выполнению, то как только в ее текст поступили все операнды, при каждой записи в регистр буфера проверяется и устанавливается признак готовности. Адрес готовой инструкции дополняет очередь "к выполнению". После выполнения инструкции вычислитель направляет результат по информации третьего адреса. Однако в связи с асинхронностью процесса в данный момент адрес отправки результата может еще быть неизвестен. Тогда результат временно сохраняется в отведенном месте текста инструкции, и инструкция остается в очереди "к выполнению", пока в ее текст

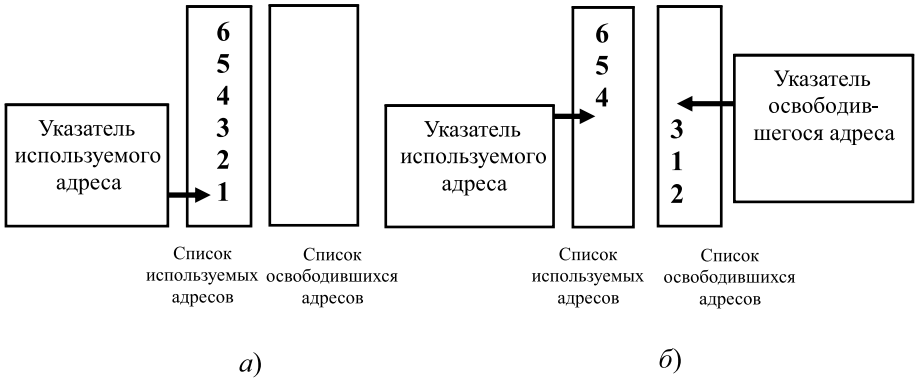

Рис. 6. Возможное состояние списков

не поступит адрес отправления. Выполненные инструкции освобождают занимаемые ими регистры. Адреса этих регистров дополняют список свободных регистров буфера данного вычислителя.

3. Процессор памяти. Принцип работы этого процессора был кратко освещен выше. Добавим, что если ОПД имеет модульную структуру и к одному модулю памяти не может быть одновременно выполнено более одного обращения, то ОЗП формируется к каждому модулю отдельно. Таким образом, заявки к каждому модулю памяти выполняются последовательно в порядке их поступления, а модули памяти работают параллельно.

Однако, если в некоторую заявку на запись по адресу не поступил код, то этот адрес блокирует выполнение последующих заявок к данному модулю с использованием этого же адреса до следующего обзора очереди к модулю. Так исключается возможное нарушение алгоритмического порядка записи и считывания при использовании адреса одной и той же ячейки памяти.

6. Рекомендации по оптимизации обмена в макетируемой ВС

Ранние попытки комплексирования многих ЭВМ для совместного решения задач привели к выводу об исключительно высоких расходах времени на обмен. Этот вывод стимулировал создание многопроцессорных вычислительных систем на общей оперативной памяти, обоснованных В. М. Бахаревым. Хотя первую такую систему CDC-6600 создал С. Крей, справедливо считать, что разработчики МВК "Эльбрус", где главным конструктором был В. С. Бурцев, были пионерами действительно практического, результативного решения, позволившего закрыть ряд важнейших государственных программ 1980-х годов.

Этот опыт в применении к макету говорит, что число запросов на обмен внутри системы следует минимизировать, и развести обмен во времени и пространстве таким образом, чтобы не создавать пиковых нагрузок на главных направлениях.

Ориентируясь на реальную разработку в будущем, в работах [1, 2] предлагается применить шесть основных независимых магистралей обмена данными, связывающих устройства вычислительной системы. Функции ряда второстепенных магистралей могут быть реализованы ими. Магистрали могут быть распределены за направлениями обмена или быть связанными коммутатором для динамического выбора менее загруженной магистрали для каждого обмена. При наличии коммутатора направления обмена между конкретными модулями ВС могут обладать плавающим приоритетом для установления "равноправия" модулей.

Для сравнения напомним, что, по-видимому, в последней модели С. Крея *Cray Superserver 6400 System* (CS6400) на базе 64 процессоров *SuperSPARC* используются четыре разделяемых магистрали.

Заключение

В основе предлагаемого макета ВС лежит утверждение: познание методов и программирование решения конкретных задач диктуют не только требование к архитектуре и системе команд, но и предлагают простейшие приемы организации взаимодействия модулей при выполнении их функций. В этом смысле вычислительная установка должна быть максимально простой [6, 7]. Разработчик, не сведущий в математических вопросах программирования, не должен "грузить" свою модель всем тем, что, как ему представляется, может понадобиться программисту (когда-нибудь) и на чем можно продемонстрировать кажущуюся эффективность своей разработки. В этом случае изделие неоправданно разрастается, требуя технологического прорыва, стоимость неоправданно растет. Проверка многих параметров при испытаниях становится недоступной.

В работе [3] методом моделирования исследована эффективность предлагаемой архитектуры при решении конкретных задач. Под эффективностью понимается значение коэффициента загрузки вычислителей. Оказывается, что этот коэффициент для единственной решаемой задачи не столь высок, что известно и для многих других параллельных архитектур. Другое дело, когда организована многоканальная обработка информации при управлении многими объектами. В этом случае много задач решаются по многим каналам одновременно. Тогда можно считать, что в сбалансированной системе с достаточно высоким коэффициентом загрузки оборудования число вычислителей должно в три...четыре раза превышать число каналов обслуживания. Поиск оптимальной архитектуры вычислительных средств приводит к особой актуальности исследования их *реальной производительности* в составе сложных управляющих систем.

Включение в состав макета процессора, выполняющего функции памяти команд, наталкивает на вывод о превращении его в *HOST*-процессор, выполняющий основные функции операционной системы и супервизора, а также представляющий данную вычислительную установку во внешних связях.

Список литературы

1. **Барский А. Б., Русаков А. Н., Хвоин Б. И.** Возможности достижения высокой скорости коммутации и внутрисистемного обмена в вычислительной системе, управляемой потоком данных // Вопросы кибернетики. Разработка и использование суперЭВМ. М., НС по комплексной проблеме "Кибернетика", 1987. С. 130—144.

2. **Барский А. Б., Шилов В. В.** Вычислительная система, управляемая потоком данных // Информационные технологии. Приложение. 2000. № 8. 24 с.

3. **Барский А. Б., Шилов В. В.** Потоковая вычислительная система: программирование и оценка эффективности // Информационные технологии. Приложение. 2003. № 7. 24 с.

4. **Игнатущенко В. В.** Организация структур управляющих многопроцессорных вычислительных систем. М.: Энергоатомиздат, 1984. 184 с.

5. **Вычислительный** комплекс "Сивуч-1". Руководство по эксплуатации. М.: АО "МЦСТ", 2014. 124 с.

6. **Бурцев В. С.** Система массового параллелизма с автоматическим распределением аппаратных средств суперЭВМ в процессе решения задачи // Юбилейный сборник трудов институтов ОИВТА РАН. Т. П. М.: ОИВТА РАН, 1993. С. 5—27.

7. **Змеев Д. Н., Климов А. В., Левченко Н. Н., Окунев А. С., Стемпковский А. Л.** Потоковая модель вычислений как парадигма программирования будущего // Информатика и ее применение. 2015. Т. 9, Вып. 4. С. 29—36.

A. B. Barsky, D. Sc., Prof., e-mail: arkbarsk@mail.ru,
D. I. Melnik, Cand. Tech. Sci., Senior Researcher, e-mail: mdi_dim@mail.ru,
NIIC (Moscow) Central Research Institute of the Air Defense Ministry of the Russian Federation

Network Technology for Designing an Asynchronous Computing System on a Common Decision Field with Self-Controlled Parallel Processing of Data

The possibility of constructing a model of a controlling two-processor computing system with a common deciding field with asynchronous data processing based on the data flow principle is investigated. The architecture of the layout is based on a compromise: The switching program executed on the processor has a traditional form, implements the traditional command system and is the subject of user development with the help of a translator. However, its commands are not executed immediately. On their basis, the computation of calculators for joint calculations is performed. The mock-up system implements the management mode of the multi-channel queuing system. On the basis of microprogramming, it is possible to develop a command system for the specialization of the control system.

The model is executed on the basis of the multi-microprocessor system "Sivuch-1", which unites 36 microprocessors "Elbrus". The purpose of prototyping is to create a tool for experimental research in the field of achieving high performance of computing facilities and special software, as well as highly stable and secure computing. Based on the results of prototyping, design proposals can be issued.

Keywords: *network technology, prototyping, data flow, processor, calculator, memory processor*

DOI: 10.17587/it.24.763-771

References

1. **Barskij A. B., Rusakov A. N., Hvoin B. I.** Vozmozhnocti dostizheniya vysokoj ckorocti kommutacii i vnutrisistemnogo obmena v vychislitel'noj sisteme, upravlyaemoj potokom dannyh (Opportunities for achieving a high commutation and communication speed in a computing system managed by the flow of data), *Voprosy kibernetiki. Razrabotka I ispol'zovanie super-EVM*, Moscow, NS po kompleksnoj probleme "Kibernetika", 1987, pp. 130—144 (in Russian).

2. **Barskij A. B., Shilov V. V.** Vychislitel'naya sistema, upravlyaemaya potokom dannyh (Computing system managed by the data flow), *Informacionnye Tehnologii*, *Prilozhenie,* 2000, no. 8, 24 p. (in Russian).

3. **Barskij A. B., Shilov V. V.** Potokovaya vychislitel'naya sistema: programmirovanie i ocenka effektivnosti (The dataflow computing system: programming and efficiency estimation), *Informacionnye Tehnologii, Prilozhenie,* 2003, no. 7, 24 p. (in Russian).

4. **Ignatushhenko V. V.** *Organizaciya struktur upravlyajushhih mnogoprocessornyh vychislitel'nyh system* (Organization of management structures of multiprocessor computing systems), Moscow, Energoatomizdat, 1984, 184 p. (in Russian).

5. ***Vychislitel'nyj*** *kompleks "Sivuch-1". Rukovodstvo po ekspluatacii* ("Sivuch-1" computer complex. Manual), Moscow, AO "MCST", 2014, 124 p. (in Russian).

6. **Burcev V. S.** Cistema massovogo parallelizma s avtomaticheskim raspredeleniem apparatnyh sredstv super-EVM v processe resheniya zadachi (The system of mass parallelism with automatic distribution of supercomputer hardware in the process of solving the problem), *Jubilejnyj sbornik tpudov institutov OIVTA RAN*, T. P. M., OIVTA RAN, 1993, pp. 5—27 (in Russian).

7. **Zmeev D. N., Klimov A. V., Levchenko N. N., Okunev A. S., Stempkovskij A. L.** Potokovaya model' vychislenij kak paradigma programmirovaniya budusshego (Flow model of computing as a programming paradigm for the future), *Informatika i eyo Primenenie*, 2015, vol. 9, iss. 4, pp. 29—36 (in Russian).