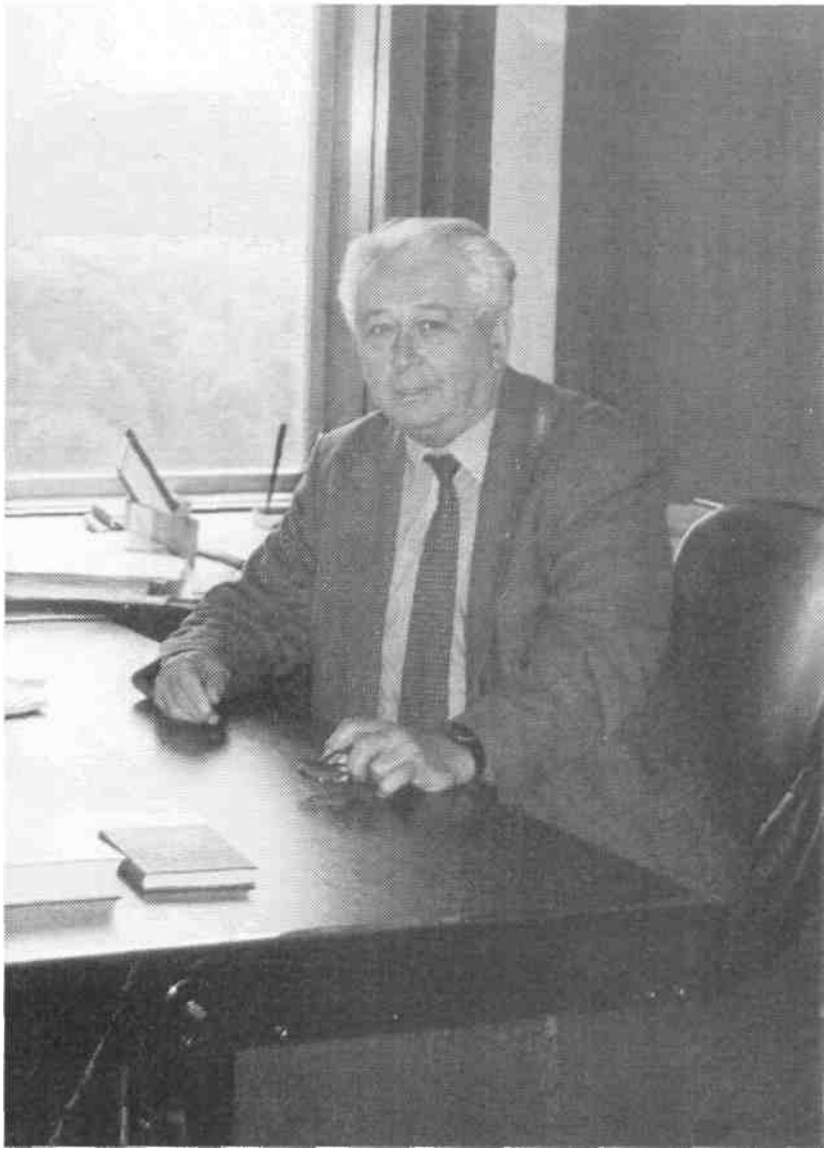
В.С. Бурцев

Параллелизм вычислительных процессов и развитие архитектуры суперЭВМ



В.С.Бурцев

Параллелизм вычислительных процессов и развитие архитектуры суперЭВМ

Москва 1997

Всеволод Сергеевич Бурцев

Параллелизм вычислительных процессов и развитие архитектуры суперЭВМ
М, 1997

В книге представлена часть трудов академика Всеволода Сергеевича Бурцева, написанных им за период с 1993 по 1996 годы. Часть работ публикуется впервые. В работах прослеживается основное направление исследований создание нетрадиционных высокопараллельных архитектур вычислительных машин и комплексов с использованием новых физических принципов, в частности, оптических и оптоэлектронных устройств.

В настоящее время средства обработки информации переживают бурное развитие. Немаловажным является тот интересный факт, что в каких бы областях науки и техники не использовались различные средства обработки информации, в конечном счете, требования к их развитию сводятся, так или иначе, к одному - увеличению производительности вычислительных средств при уменьшении потребляемой мощности и повышению надежности. Этим, наверное, объясняется то обстоятельство, что архитектурные и схемотехнические решения, принятые в суперЭВМ - ЭВМ, обладающих наибольшей производительностью на фиксированный момент времени, определяют передний фронт развития этой отрасли.

Увеличение производительности обработки информации, в основном, достигается двумя путями: первый - увеличением быстродействия элементной базы, включая конструкцию, и второй - архитектурными решениями построения средств обработки информации. Первый путь практически исчерпал все возможности и дальнейшее продвижение в этом направлении требует открытия новых физических принципов обработки информации.

Второй путь, в основе которого лежит распараллеливание процессов обработки информации на всех уровнях решения задачи, широко использовался на всех этапах развития информационно-вычислительных средств:

- переход от последовательного счета к параллельному;
- параллельная работа основных устройств ЭВМ;
- введение пакетного режима работы и режима разделения времени;
- мультиплексный режим обработки данных;
- конвейерный режим;
- многопроцессорные и многомашинные комплексы;
- режим управления потоками данных и т. д.

Второй путь повышения производительности обработки информации на настоящее время далеко не исчерпал себя и с успехом будет развиваться в перспективных суперЭВМ.

Можно с уверенностью сказать, что развитие архитектуры суперЭВМ идет в направлении создания условий для выявления и реализации параллельных процессов на всех уровнях взаимодействия человека с суперЭВМ.

Этим обстоятельством и объясняется название данной книги, в которой собраны статьи, написанные в 1993-1996 г.г., опубликованные в различных журналах и сборниках, а также вновь подготовленные к печати. В них прослеживаются вышеприведенные тенденции развития вычислительной техники и предлагаются методы их реализации на современном этапе.

Сборник составлен на базе опыта многолетней работы совместно с большим коллективом сотрудников, из которых особую благодарность автор выражает к.т.н. Хайлову И.К., Сызько Э.В., д.ф.м.н. Подшивалову Д.Б.,
д.т.н. Перекатову В.И., Козлову Л.А., д.ф.м.н. Оленину А.С., к.т.н. Никольской Ю.Н., к.ф.м.н. Тарасенко Л.Г., д.т.н. Федорову В.Б., д.т.н. Торчигину В.П., д.ф.м.н. Захарову С.М., к.т.н. Шабанову Б.М., к.ф.м.н. Степанову А.М., к.т.н. Березко А.М., к.ф.м.н. Фетисову Н.С., к.ф.м.н. Копейкину А.Б., к.т.н. Кострюкову В.А., к.ф.м.н. Игнатову В.В., к.т.н. Гуснину С.Н., Андрееву А.В., Ткачеву В.А., Твердохлебову М.Н. и многим другим.

Автор чрезвычайно признателен Никитину Ю.В., Давыдовой Г.Н.,Суима Е.А., Зотовой И.С., Табаковой Л.Н., Костеревой И.Н., Николаевой В.А., Шашуриной Т.П., которые в рекордно короткий срок подготовили сборник к печати.

Автор будет благодарен за мнения, высказанные об этой книге. Если читатели сочтут ее интересной, будет сделана попытка подготовить сборник по анализу архитектур более ранних разработок суперЭВМ советского периода.

В.С.Бурцев

Значение создания ENIAC в развитии информационно-вычислительных и управляющих систем России

(Доклад на юбилейной сессии ЮНЕСКО по поводу пятидесятилетия создания первой ЭВМ ENIAC. Москва, 1996г.)

В.С.Бурцев

Аннотация

Отмечается, что создание первой ЭВМ дало мощный импульс в развитии информационной техники и изменило методы разработки сложных управляющих систем и объектов. Приводятся примеры из истории развития вычислительной техники в России, большинство из которых публикуются впервые.

Пятьдесят лет назад в университете Пенсильвании (США) был создан первый в мире электронный компьютер дискретного действия с автоматической обработкой информации в соответствии с заложенной в него программой. Значение этого события трудно переоценить, так как в результате был дан колоссальный импульс развитию такого свойства современного общества, как его глобальная информатизация. Это событие повлекло за собой коренной пересмотр методов научных исследований и разработок, в результате чего стало возможным создание новых сложных комплексов, определяющих научно-технический уровень развития общества. К числу достижений в этой области можно отнести: создание новых воздушных и морских лайнеров; бурное развитие атомной энергетики и освоение космоса; создание сложных радиолокационных и управляющих систем и многие другие направления прогресса. По ряду сложившихся обстоятельств этот импульс научно-технического развития особенно ощутим был в Советском Союзе - все основные передовые разработки и проекты уже с 1956 года базировались на использовании дискретной вычислительной техники, что не могло не стимулировать ее быстрое развитие. Процесс развития вычислительной техники лучше всего можно проследить по работам специалистов школы академика С.А.Лебедева, основоположника и

первопроходца вычислительной техники в Советском Союзе. Работы, проводимые школой С.А.Лебедева, не могли быть опубликованы до настоящего времени, так как основные и наиболее интересные из них выполнялись по заказам оборонных ведомств и были связаны с системами противоракетной и противосамолетной обороны, освоением космоса, созданием ядерных вооружений. После выхода целого ряда книг и статей по этим направлениям науки и техники [1,6,7,8] стало возможным проследить развитие вычислительной техники в нашей стране. Настоящее сообщение делает попытку в какой-то мере заполнить определенный пробел в истории развития отечественной вычислительной техники.

Первая электронная вычислительная машина дискретного действия, получившая название Малая электронная счетная машина (МЭСМ), была создана под руководством С.А. Лебедева в 1950 году в Институте электротехники АН Украины. На этой машине С.А.Лебедев проверил многие принципы организации вычислительного процесса при работе в двоичной системе исчисления. Данная работа послужила хорошим экспериментом, позволившим ему уже через три года создать Быстродействующую электронную счетную машину (БЭСМ) для расчета многих задач экспериментальной физики и ряда других задач, которые ранее из-за большой трудоемкости счета решить было невозможно. Если МЭСМ имела производительность 50 оп/с, последовательное 17-разрядное арифметическое устройство (АУ) с фиксированной запятой, всего четыре арифметические операции и одну команду управления при емкости оперативного запоминающего устройства (ОЗУ) в тридцать одно слово, то БЭСМ в 1953 году обладала производительностью в 12 тысяч оп/с, имела параллельное 39-разрядное АУ с плавающей запятой, емкость ОЗУ 1024 слова и выполняла 32 различные операции. БЭСМ имела достаточно развитую систему внешней памяти: барабан, ленту, устройства ввода с перфокарт и устройство вывода на печатающее устройство. В БЭСМ существовали специальные команды изменения центрального и местного управления командами ИЦУК и ИМУК, обеспечивающие переход к подпрограммам с возвратом в исходную точку программы. В то же время управление БЭСМ было построено по последовательной схеме организации работы всех основных устройств - АУ, УК (управление командами), ОЗУ, барабана, ленты, печати и перфокарт.

Рис.1. Временная работа БЭСМ.

NK - время обращения за командой, A1, A2 - время приема двух операндов, АУ - время работы арифметического устройства, A3 - время записи результата, МБ - время обращения к магнитному барабану, A_1, ... A_n - время обращения к памяти по записи или считыванию (пслов).

Согласно жесткой последовательности работы устройств БЭСМ (Рис. 1), сначала из ОЗУ считывался код команды, которая организовывала необходимые обращения к ОЗУ за считыванием первого и второго числа, выдавала необходимую

временную диаграмму для выполнения операции на АУ, записывала результат и считывала следующую команду. При обращении к ленте, барабану или внешним устройствам, центральное управление обеспечивало все операции поиска информации, ее считывания или записи, для чего использовалось АУ.

В 1953-1956 годах в Институте точной механики и вычислительной техники (ИТМ и ВТ) АН СССР проводились работы по автоматическому съему данных и сопровождению целей с радиолокационной станции (РЛС), которые открыли путь к созданию радиолокационных и ракетных комплексов на новой информационно-вычислительной основе. В 1955 году на макете радиолокационной станции обзорного действия был проведен эксперимент одновременного сопровождения нескольких реальных целей (самолетов) при опережающем расчете их траектории. Выдача координат осуществлялась в дискретном цифровом виде. В 1956 году этот эксперимент был повторен на серийной станции П-30 с посылкой оцифрованного сигнала управления на самолет перехвата целей. Работы проводились с использованием специальных ЭВМ "Диана I" и "Диана II". Оцифровку и селекцию данных, построение траекторий целей для их сопровождения осуществляла "Диана I", а решение задачи перехвата и выдачу команд управления "Диана II". Эти работы позволили в 1960 году построить радиолокационный комплекс наведения противоракеты на баллистическую ракету противника с целью ее уничтожения осколочным зарядом с диапазоном отклонения от цели не более 25 метров. Для решения этой проблемы потребовалось создать высокопроизводительную вычислительную сеть. Производительность центральной ЭВМ этой сети достигала 40 тысяч оп/с при объеме ОЗУ в 4096 40-разрядных слов. Создание этой машины под названием М-40 было закончено в 1958 году. Для достижения столь высокой производительности были существенно пересмотрены принципы организации системы управления ЭВМ.

Каждое устройство машины УК, АУ, ОЗУ, УВУ (управление внешними устройствами) получили автономное управление, что позволило реализовать их параллельную во времени работу. С этой целью был создан мультиплексный канал обращения к ОЗУ со стороны УК, АУ и УВУ.

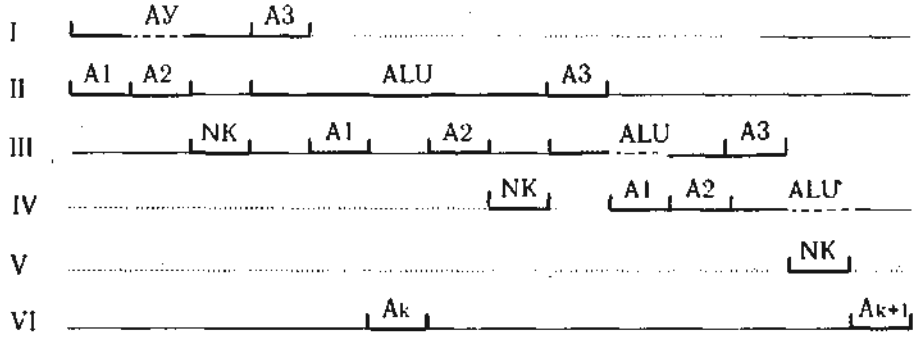


Рис.2. Временная диаграмма работы М-40.

I - V - временные диаграммы выполнения арифметических операций, VI - временная диаграмма выполнения операций обращения к внешним устройствам, A_1, A_2 - времена обращений к ОЗУ за операндом, A_3 - время записи результата, NK - время считывания команды, АУ - время работы АУ.

Эти устройства рассматривались как самостоятельно работающие процессоры, обращающиеся к общему ОЗУ, то есть как машина, фактически представлявшая собой многопроцессорный комплекс.

Согласно временной диаграмме М-40 (Рис.2), обращение к памяти УК (А, А2, А3 и NK) и работы управления внешних устройств (УВУ), включая систему передачи данных, процессор ввода-вывода данных (ПВВ), происходят, как правило, на фоне работы АУ, УК, УВУ и ПВВ.

В экспериментальном комплексе противоракетной обороны (ПРО) эта машина осуществляла обмен информацией по пяти дуплексным одновременно и асинхронно работающим радиорелейным каналам связи с объектами, находящимися от нее на расстоянии от 100 до 200 километров; общий темп поступления информации через радиорелейные линии превышал 1 МГц (Рис.3).

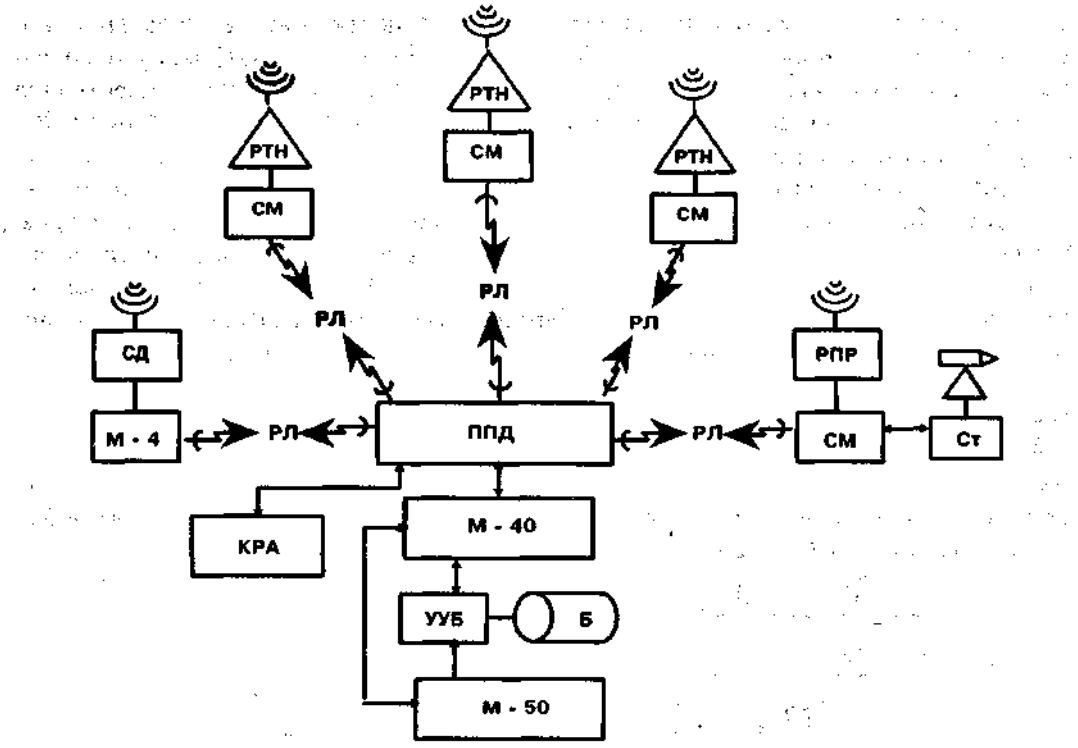


Рис.3. Вычислительная сеть экспериментальной системы ПРО.

РТН - радиолокаторы точного наведения, СМ - специальные вычислительные машины, СД - станция дальнего обнаружения, М-4 - электронная вычислительная машина М-4, РПР - радиолокатор противоракеты (передача сигналов управления на противоракету), СТ - стартовая установка противоракет, ППД - процессор приема и передачи данных, М-40 и М-50 - универсальные электронные вычислительные машины М-40 и М-50, Б - запоминающее устройство на магнитном барабане, УУБ - устройство управления барабаном, КРА - контрольно-регистрирующая аппаратура, РЛ - радиорелейные линии.

Проблема обмена информацией с асинхронно работающими объектами была решена с помощью процессора ввода-вывода (ПВВ), работа которого основывалась

на принципе мощного мультиплексного канала, имеющего свою память, доступную для всех каналов.

Одновременно с проведением боевой работы М-40 осуществляла запись на внешнее запоминающее устройство (барабан) экспресс-информации, которая обрабатывалась на аналогичной ЭВМ М-50 (модернизация М-40, обеспечивающая работу с плавающей запятой). Боевые пуски по всем направлениям входа и выхода сопровождались записью информации на магнитные ленты контрольно-регистрирующей аппаратуры (КРА). Это давало возможность в реальном масштабе времени "проигрывать" и анализировать каждый пуск, для чего ЭВМ М-40 и М-50 имели развитую систему прерываний. Быстрому вводу вычислительной сети противоракетной обороны (ПРО) в значительной степени помог задел, который был создан в институте при разработке ЭВМ М-20. Проектирование этой ЭВМ было начато в 1955 году и завершено в 1958 году. Машина имела 20 тысяч оп/с, ОЗУ емкостью 4096 45-разрядных слов, арифметическое устройство с плавающей запятой, развитую систему внешних устройств, печатающее устройство с автономным управлением. На фоне работы арифметического устройства осуществлялась выборка следующей команды. Обеспечивалось автономное управление вводом-выводом (работа с печатью и перфокартами с использованием автономной памяти). Было сохранено централизованное последовательное управление АУ, ОЗУ и УК, за счет чего достигнута экономия оборудования при определенном проигрыше в скорости вычислений (Рис.4).

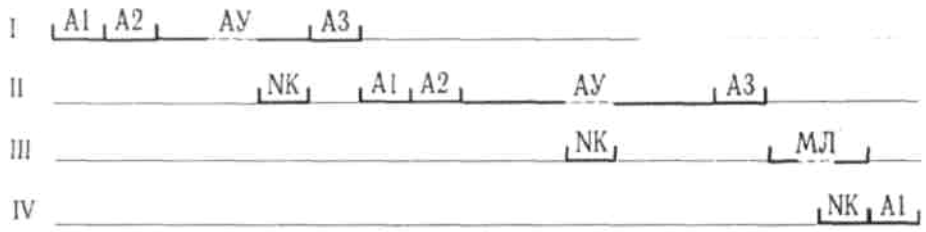


Рис.4. Временная диаграмма работы М-20.

/, //, IV - временные диаграммы выполнения арифметических операций, III - временная диаграмма выполнения операции обращения к магнитной, ленте (МЛ), А1, А2 - времена обращений к ОЗУ за операндом, А3 - время записи результата, NK - время работы АУ.

ЭВМ М-20 хорошо зарекомендовала себя в эксплуатации и более десяти лет была основной ЭВМ общего назначения в СССР. М-20 была модернизирована (выполнена на полупроводниковой элементной базе), после чего она серийно выпускалась под названием БЭСМ-4.

Опыт эксплуатации экспериментального комплекса ПРО показал, что его вычислительные средства можно рассматривать как мозг всей системы; малейшие отклонения от нормы их функционирования приводят к нарушению работы всего комплекса, что может вызывать опасные ситуации. Поэтому при создании вычислительных средств на полупроводниковых элементах для боевого комплекса особое внимание было уделено устойчивости их работы при сбоях и отказах.

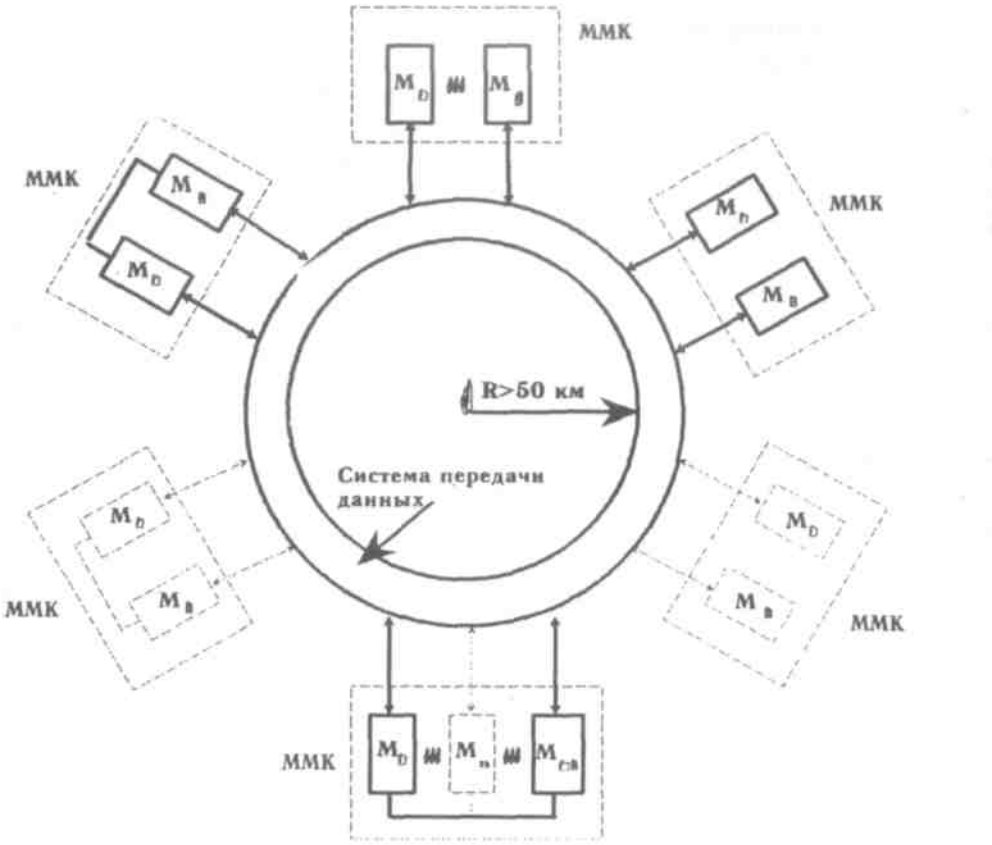


Рис.5. Вычислительная сеть ПРО.
ММК - многомашинный комплекс.

Вычислительная сеть системы ПРО (Рис.5) имела протяженность несколько сот километров. Она состояла из вычислительных комплексов, каждый из которых был построен из идентичных боевых ЭВМ, обладающих полным пооперационным аппаратным контролем. Резервирование в комплексе обеспечивалось на уровне машин.

На Рис.6 показана структурная схема центрального 12 машинного комплекса системы ПРО со скользящим резервированием. На десять функционально работающих машин (Ml - M10) предусматривалось две машины (M11- M12) для горячего резервирования, которые работали в режиме "подслушивания" и были готовы в течение нескольких десятков миллисекунд заменить любую из вышедших из строя ЭВМ. Сигнал неисправности ЭВМ вырабатывался аппаратно системой пооперационного контроля каждой ЭВМ и посылался в систему прерывания всех машин. По межмашинному обмену наряду с данными боевого цикла передавалась необходимая экспресс-информация для ЭВМ, находящейся в резерве. В этом комплексе шесть ЭВМ (M_1 - M_6) решали задачу обнаружения целей по данным радиолокатора дальнего действия и построения их траекторий. Четыре ЭВМ (M_7 - M_{10}) решали задачи управления системой, включая распределение целей по стрельбовым комплексам.

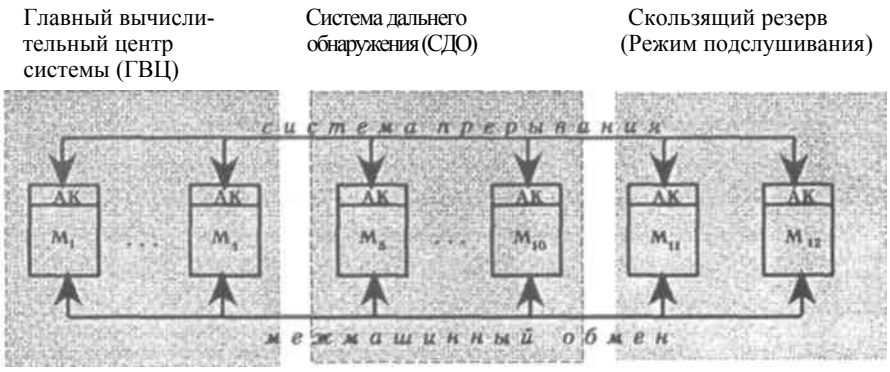


Рис.6. Система сквозного резервирования на уровне машин.
M_1 - M_{12} - универсальные ЭВМ 5Э926, АК - система аппаратного контроля.

Эти ЭВМ, под названием 5Э526, имели производительность 0,5 млн. оп/с над числами с фиксированной запятой и ОЗУ объемом 32 тысячи 48 разрядных слов. Все основные устройства ЭВМ имели автономное управление, а управление внешними устройствами осуществлялось процессором передачи данных, имеющим довольно развитую специализированную систему команд. Серийный выпуск этих машин для управления различными стационарными средствами вооружения был начат с 1966 года. Машина была модернизирована в части введения арифметики с плавающей запятой и мультипрограммного режима. Модернизированная ЭВМ имела название 5Э51 и серийно выпускалась с 1967 года для построения мощных вычислительно-информационных центров повышенной надежности. Благодаря автономной работе ее основных устройств и, в первую очередь, процессора ввода-вывода, на базе общего ОЗУ эти машины успешно использовались при создании многомашинных комплексов с единой внешней памятью, состоящей из большого количества барабанов, дисков и лент. Структурная схема одного из таких комплексов, Центра контроля космического пространства (ЦККП), показана на Рис.7.

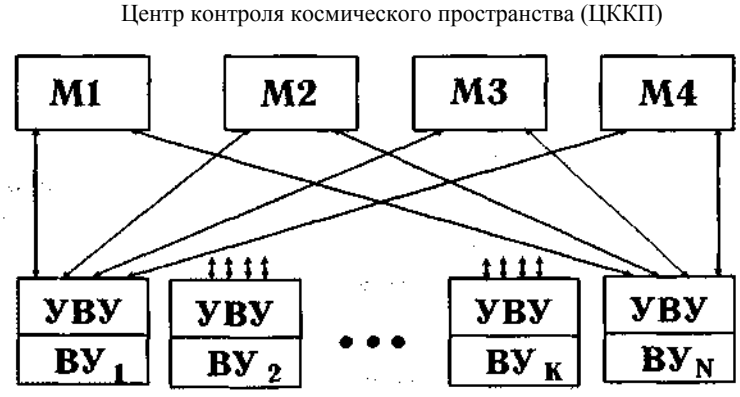


Рис.7. Многомашинная работа на единую внешнюю память. *М1 - М4 - ЭВМ 5Э51, УВУ - устройства управления внешней памятью и устройствами ввода-вывода, ВУ - внешние устройства (внешняя память, управление каналами обмена и устройствами ввода-вывода).*

В это же время, в институте на аналогичной полупроводниковой элементной базе создается универсальная ЭВМ БЭСМ-6 (Рис.8), которая в течение двадцати лет широко использовалась как основная быстродействующая ЭВМ в различных вычислительных комплексах Советского Союза.

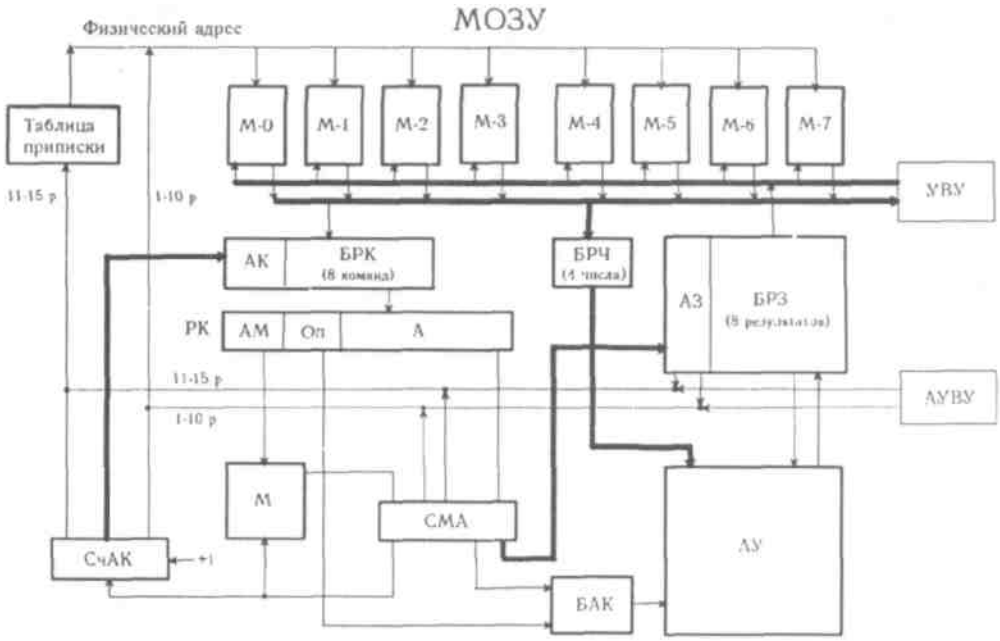


Рис.8. Блок-схема центральной части машины БЭСМ-6.

А - адрес оператора, АЗ - буфер адресов записи, АК - буфер адреса команд, АМ - адрес модификатора, АУВУ - адрес устройства управления внешними устройствами, БАК - базовый адрес команды, БРК - буферные регистры команд, БРЗ - буфер регистров записи результата, БРЧ - буферные регистры чисел, М - регистры модификаторов, МОЗУ - магнитное запоминающее устройство, М-0 - М-7 - модули оперативной памяти, РК - регистр команды, СМА - сумматор адреса, СчАК - счетчик адреса команд.

За счет оригинальной схемотехники и конструкции БЭСМ-6 имела производительность 1млн. оп/с, объем ОЗУ 32 тысячи 50-разрядных слов, АУ параллельного типа с плавающей запятой. Существенного увеличения производительности БЭСМ-6 удалось достичь за счет использования конвейерного принципа организации вычислительного процесса и введения интерливинга в модульную память (интерливинг на 8 модулей). Для записи чисел были использованы регистры с ассоциативной выборкой (типа кэш). Эффективная работа с математической памятью обеспечивалась при аппаратной поддержке защиты в многопрограммном режиме работ. С этой же целью был введен специальный привилегированный режим работы для отдельных блоков операционной системы. В БЭСМ-6 сохранился централизованный принцип управления основными устройствами, что существенно сдерживало организацию параллельной работы внешних устройств и затрудняло создание комплексов с общей внешней памятью и работу с каналами связи. Чтобы обеспечить комплексирование БЭСМ-6, на той же элементно-конструктивной основе был разработан комплекс

АС-6, позволявший объединять ОЗУ нескольких машин в общую память высокопроизводительными каналами с пропускной способностью в 1,3 млн. слов в секунду каждый. В комплексе предусматривался второй уровень коммутации внешних устройств и каналов связи, для чего использовались автономные системы управления каналами связи и внешними устройствами; эти системы сопрягались со стандартным каналом связи внешних устройств производительностью 1,5 Мб/с. Управление комплексом осуществлялось новым процессором АС-6 с отличной от БЭСМ-6 системой команд. Система АС-6 была реализована в трех взаимно дублирующих вычислительных центрах обработки космической информации.

Таким образом, С.А.Лебедев как главный конструктор всех разработок умело использовал финансовые и приоритетные возможности военных заказов, разрабатывая параллельно более дешевые высокопроизводительные ЭВМ гражданского применения.

Так, в связи с необходимостью дальнейшего развития высокопроизводительных ЭВМ, уже в 1967 году встал вопрос о переходе на новую конструктивно-технологическую базу с использованием интегральных схем (ИС) и печатных плат. Это требовало переоснащения Института и работающих с ним заводов, освоения новых методов автоматизированного проектирования для размещения ИС, трассировки многослойных печатных схем и межблочного монтажа, создания тестово-диагностических программ.

Такое переоснащение можно было осуществить, только взяв заказ на разработку серийной возимой вычислительной системы для противосамолетного комплекса С-300. По этой разработке ИТМ и ВТ впервые был заказчиком серии интегральных схем (ТТЛ) для цифровой техники в Министерстве электронной промышленности и получил достаточно средств на переоснащение заводов и Института. Так как комплекс требовал повышенной устойчивости при работе в сложных климатических и вибрационных условиях, он был реализован по крупномодульному принципу с полным автономным аппаратным контролем каждого модуля. Резервирование комплекса осуществлялось не на машинном уровне, а на уровне модулей основных устройств ЭВМ, что значительно эффективнее.

ЭВМ 5Э26 (Рис.9) имела суммарную производительность трех процессоров 1 млн. оп/с, арифметико-логическое устройство (АЛУ) с фиксированной запятой с шириной слова в 35 разрядов, ОЗУ емкостью 32 Кбит. В дополнение к ОЗУ имелась память команд (ПК) объемом 64 Кбит. ПК была реализована на бибсах (ферритовых сердечниках с двумя взаимно перпендикулярными отверстиями), работала без разрушения считываемой команды и обеспечивала хранение информации без расхода энергии. Общий объем ЭВМ составлял менее 2,5 $м^3$, а потребляемая мощность имела значение порядка 5 кВт. Реализация вычислительного комплекса с такими параметрами стала возможной только при использовании ИС и конструкции на многослойных печатных платах. Комплекс 5Э26, несмотря на достаточно низкую надежность ИС на первом этапе их производства ($\lambda > 10^{-6}$), обеспечивал функциональную надежность не ниже 0,99 в самых тяжелых условиях большого перепада температур, повышенной влажности и тряски при резервировании АУ и УК два из трех и стопроцентном резервировании ПК, ОЗУ и УВУ.

Рис.9. Система резервирования на уровне модулей устройств возимой ЭВМ 5Э26 *АУ - арифметическое устройство, УУ - устройство управления, АК - пооперационный аппаратный контроль, КМ - разнесенный (по модулям) центральный коммутатор, ПК - неразрушаемая память команд, УВУ - устройство управления внешними устройствами, ОЗУ - оперативное запоминающее устройство.*

При этом удалось реализовать такую скорость замены дефектных устройств, что практически на всей временной диаграмме функционирования комплекса сбои и отказы устройств не влияли на нормальное продолжение его боевой работы.

Естественным продолжением работ по многопроцессорной архитектуре ЭВМ было создание на этом принципе комплекса вычислительных средств, обладавшего предельной производительностью при частотных характеристиках элементов, которые обеспечивала технология того времени. В отличие от машины 5Э26, где многопроцессорный модульный принцип был использован, в основном, для обеспечения высоких надежностных характеристик, он, наряду с высокими показателями надежности в многопроцессорном вычислительном комплексе МВК "Эльбрус-2", обеспечил предельную производительность. Производительность многопроцессорной системы, в основном, ограничивается двумя факторами: пропускной способностью коммутатора между процессорами и ОЗУ и сложностью организации корректной работы сверхоперативной кэш памяти. Трудности, возникающие при решении этих двух проблем, существенно увеличиваются с ростом количества процессоров. Пропускная способность коммутатора "Эльбрус-2" достигала 2Гбайт/с. Корректность работы сверхоперативной памяти была обеспечена путем ее разбиения на несколько частей, каждая из которых имела свой алгоритм работы в соответствии с выполняемой функцией. Так, в МВК "Эльбрус-2" имеется: сверхоперативная память команд, массивов, локальных данных, безадресный буфер быстрых регистров, построенный по принципу потока данных и буфер глобальных данных. Корректность работы буфера глобальных данных с ОЗУ вызывает те же проблемы в многопроцессорном комплексе, которые возникают в сверхоперативной КЭШ. В МВК "Эльбрус-2" реализована схема корректности работы этого буфера, которая обеспечивает корректную работу комплекса, практически не замедляя ее вне зависимости от числа процессоров [4]. Аналогичные схемы, используемые в современных комплексах фирмы Hewlett Packard (SPP-2000) и Silicon Graphics, существенно уступают по эффективности схеме МВК "Эльбрус-2". МВК "Эльбрус-2" создавался в два этапа:

- на первом этапе отрабатывались новые архитектурные принципы, включая программное обеспечение;

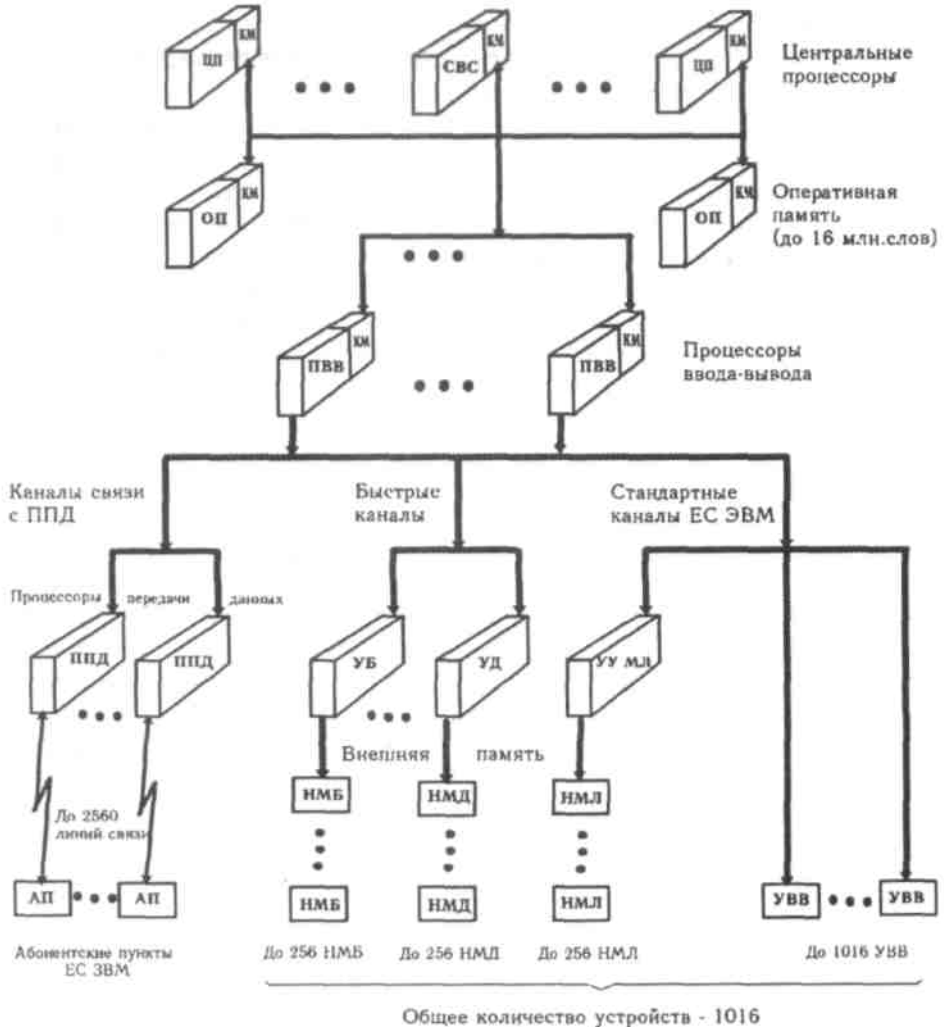


Рис.10. Структура МВК "Эльбрус".

ЦП - центральный процессор, ОП - оперативная память, КМ - коммутатор, обеспечивающий доступ каждого ЦП к каждому модулю ОП и ПВВ, ПВВ - процессор ввода-вывода, ППД - процессор передачи данных, УБ - устройство управления барабанами, УД - устройство управления дисками, УУ МЛ - устройство управления магнитными лентами, НМБ - накопитель на магнитных барабанах, НМД - накопитель на магнитных дисках, НМЛ - накопитель на магнитной ленте, УВВ - устройства ввода-вывода.

- на втором этапе наряду с принципами архитектуры отрабатывалась новая конструкторско-технологическая база.

На первом этапе был реализован 10-процессорный комплекс "Эльбрус-1" производительностью в 15 млн. оп/с на элементно-конструкторской базе 5Э26 на ТТЛ элементах с задержкой 10-20 нс на вентиль. На втором этапе был создан МВК "Эльбрус-2", производительностью 120 млн. оп/с и с объемом ОЗУ 160 Мбайт, построенный на элементной базе типа Motorola 10000 с задержкой 2-3 нс на вентиль.

МВК "Эльбрус" (Рис.10) построен по модульному принципу и в зависимости от комплектации может включать необходимое количество центральных процессоров (1-10), модулей оперативной памяти (4-32), процессоров ввода-вывода (ПВВ) (1-4), устройств внешней памяти (барабанов, дисков, магнитных лент), процессоров передачи данных (ППД) (1-16) и устройств ввода-вывода, подключенных либо непосредственно к ПВВ, либо через линии передачи данных посредством ППД. Каждый компонент комплекса, включая разнесенные по ним узлы центрального коммутатора, имеет стопроцентный аппаратный контроль и при появлении хотя бы одиночной ошибки в ходе вычислительного процесса выдает сигнал неисправности. По этому сигналу операционная система через аппаратно реализованную систему реконфигурации исключает неисправный модуль из работы.

Отключенный модуль попадает в ремонтную конфигурацию, в которой посредством тест-диагностических программ и специальной аппаратуры ремонтируется, после чего может быть включен операционной системой в рабочую конфигурацию.

Описанная структура позволяет осуществить резервирование на уровне однотипных модульных устройств. Время подключения резервного модуля не превосходит 0,01 сек, что обеспечивает бессбойную работу комплекса с заданной надежностью для всех боевых систем.

К особенностям МВК "Эльбрус-2", кроме описанных выше, можно отнести следующие:

- система команд эффективно реализует автокод, являющийся языком высокого уровня;
- аппаратно поддержаны часто встречающиеся программные конструкции языков высокого уровня;
- осуществлена обезличенная работа однотипных модулей процессоров, ПВВ, ППД;
- аппаратно реализована в ПВВ работа диспетчера внешних объектов;
- система команд и математическое обеспечение ППД позволяют достаточно просто адаптироваться к различным вычислительным сетям и внешним объектам;
- в качестве одного из процессоров может быть подключен любой спецпроцессор (если таким спецпроцессором является процессор, то достигается производительность, более чем в шесть превышающая производительность БЭСМ-6).

Аванпроект МВК "Эльбрус" был выполнен в 1970 году, МВК "Эльбрус-1" сдан Госкомиссии в 1980 году, а МВК "Эльбрус-2" в 1985 году. Оба комплекса серийно выпускались более 15 лет.

За последние 10 лет в России специалистами школы академика С.А.Лебедева было разработано много интересных проектов, внесших существенный вклад в развитие архитектуры суперкомпьютеров, однако по тем или иным причинам они не были реализованы в серийном производстве. К ним надо отнести: векторный процессор МВК "Эльбрус-2", модульный конвейерный процессор, МКП ССБИС, оптическую сверхвысокопроизводительную вычислительную машину (ОСВМ) РАН и ряд других.

В заключение можно сказать, что по целому ряду архитектурных и схемотехнических решений российская школа суперЭВМ нисколько не отстает от

зарубежной. Поэтому научно-техническое сотрудничество с зарубежными коллегами в области создания суперЭВМ было бы чрезвычайно полезным для обеих сторон.

Литература

1. Б.Н.Малиновский. История вычислительной техники в лицах.
2. Вычислительные машины с нетрадиционной архитектурой. Супер-ВМ. Сборник научных трудов ВЦКП. вып.2. Москва 1994.
3. Академия наук УССР. Библиография ученых Украинской ССР. Сергей Алексеевич Лебедев. Киев, Наукова Думка 1978.
4. В.С.Бурцев Принципы построения многопроцессорных вычислительных комплексов "Эльбрус". Доклад на научно-техническом семинаре Многопроцессорные вычислительные комплексы. Москва 21-22 ноября 1977. Препринт № 1 за 1977.
5. Л.Н.Королев, В.А.Мельников. Об ЭВМ БЭСМ-6. Управляющие системы и машины, 1976, 6.
6. Голубев О.В., Каменский Ю.А., Минасян М.Т., Пупков Б.Д. Российская система противоракетной обороны (прошлое и настоящее - взгляд изнутри). Техноконсалт, Москва, 1994.
7. Кисунько Г.В. Секретная зона. Москва, Современник, 1996.
8. Бакулев П.А., Сотский Н.М., Горохов Ю.И. О некоторых работах в СССР по внедрению средств цифровой автоматики в радиолокацию и в радиотехнические комплексы управления летательным аппаратом.
9. Бурцев В.С. Новые подходы к оценке качества вычислительных средств. (в данной книге)

О необходимости создания суперЭВМ в России

Бурцев В.С.

Аннотация

В статье приведены требования, предъявляемые к суперЭВМ при решении сложных задач. Дана оценка возможностей современных суперЭВМ. Указаны основные причины, сдерживающие достижение требуемой производительности. Указано на особые условия развития суперЭВМ в нашей стране. Сделан вывод о необходимости создания суперЭВМ в России на зарубежной элементной базе.

В последнее время, очевидно, под воздействием быстро развивающихся технологических возможностей, которые привели к созданию мощных однокристальных микропроцессоров (Альфа, Power PC, Pentium, Intel 860 и др.), все чаще высказывается мнение, что время суперЭВМ уже прошло, что все задачи можно решить на современных PC и рабочих станциях, объединенных в единую локальную сеть. Самое странное, что такие мнения высказываются в академических кругах и в газете "Поиск". Некоторые ученые в своих высказываниях идут даже дальше и говорят о том, что больших задач, для которых необходимы суперЭВМ, в академии нет. Они пытаются объяснить этот "феномен" тем, что математические методы решения сложных задач, разработанные нашими математиками, позволили решать эти сложные задачи на рабочих станциях.

Все это заставило меня написать эту статью и попытаться ответить на следующие вопросы:

- каковы современные требования, предъявляемые к суперЭВМ, исходя из необходимости решения проблемных задач первостепенной важности;
- каковы возможности современных PC, рабочих станций и суперЭВМ в настоящее время;

- можем ли мы создать суперЭВМ, соответствующую современному мировому уровню или даже превышающую по характеристикам зарубежные, и надо ли этим заниматься.

Для определения основных требований, предъявляемых к суперЭВМ сложными проблемными задачами, воспользуемся данными, приведенными американским ученым Гарри Джонсоном из суперкомпьютерного центра в Северной Каролине, США [1], и мнением наших ведущих ученых, работающих в различных направлениях исследований и широко использующих математические методы моделирования. На Рис.1 в координатах производительность-память даны требования к суперЭВМ для различных задач. Производительность посчитана исходя из требования 15-минутного решения задачи. Американские данные помечены квадратами. Требования к аналогичным задачам, которые привели наши математики, того же порядка (они обозначены треугольниками) Пунктирной линией показана область, которую перекрывают наиболее производительные суперЭВМ по теоретической пиковой производительности - это Cray T3D, nCUBE-3 и CM-5, ориентированные на решение задач уравнений математической физики.

Машины с такой пиковой производительностью (максимальной комплектации) еще не созданы, поэтому сплошной чертой показана производительность реально работающих машин, полученная на фрагментах тестовой задачи [2]. На этом же графике показана область, которую перекрывают универсальные, наиболее производительные многопроцессорные суперЭВМ конвейерного типа Cray Y-MP. Из приведенной на Рис.1 информации можно сделать вывод, что суперЭВМ в настоящее время не удовлетворяют предъявляемым к ним требованиям по производительности и объему памяти на несколько порядков. Однако, не стоит делать неправильного вывода о том, что задачи по этим направлениям в настоящее время не решаются. Анализ данных, приведенных в [1], показывает, что по всем из обозначенных на рисунке направлениям ведутся работы на существующих суперЭВМ, однако выполняемые ими задачи не решают поставленных проблем в полном объеме. Решаемые задачи развиваются в соответствии с развитием возможностей суперЭВМ. По каждому направлению вырабатывается своя линия развития задачи, которая во многом определяется возможностями суперЭВМ на каждый момент времени.

На Рис.2 в качестве примера показаны три направления исследований, каждое их которых обозначено на рисунке соответствующим значком. Из рисунка видно, что вычислительные средства в настоящее время не удовлетворяют требованиям этих задач (незаштрихованные значки). В то же время многие задачи по направлениям этих исследований решаются на существующих суперЭВМ (заштрихованные значки).

В той же плоскости координат на Рис.3 показаны возможности двух типов вычислительных систем:

- многопроцессорных комплексов с универсальными скалярно-векторными конвейеризованными процессорами; эти структуры (помеченные на рисунке треугольниками), где каждый процессор имеет прямой доступ к каждому модулю общей памяти, называют системами с распределяемой памятью в отличие от многопроцессорных систем с распределенной памятью;

- многопроцессорных систем с распределенной по процессорам памятью (Cray T3D, транспьютерные системы Парситек, nCUBE-2 и СМ-3 (помечены квадратами); в некоторых работах их относят к многомашинным комплексам [3].

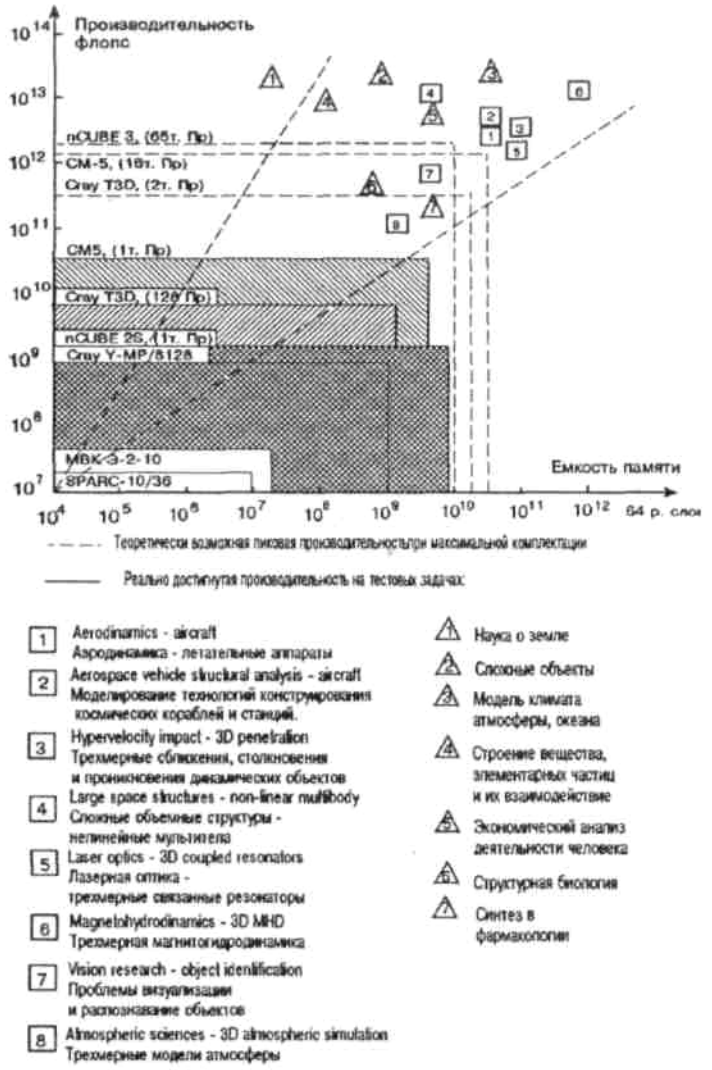


Рис.1

Как уже отмечалось, даже самые лучшие суперЭВМ настоящего времени по производительности на несколько порядков не отвечают требованиям больших задач.

Каковы те принципиальные причины, которые не позволяют построить суперЭВМ требуемой производительности? Если совсем коротко, то они ограничены отводом тепла от полупроводниковых схем. Принципиально жидкостное охлаждение позволяет отвести 20 Вт с одного квадратного сантиметра. Следовательно,

чем меньше выделяет энергии схема на одно логическое срабатывание Аэ, тем меньше выделяется мощности Рэ=Аэ/tэ при том же быстродействии, где tэ - время срабатывания элемента. Следовательно, в одном и том же объеме можно сосредоточить большее количество логических элементов и элементов памяти. Если объем растет, растут потери во времени на связи между элементами tсв и времена tэ становятся соизмеримы, а иногда и много меньше, чем tсв. Поэтому, чем выше производительность элемента, тем меньший объем должна иметь вычислительная система.

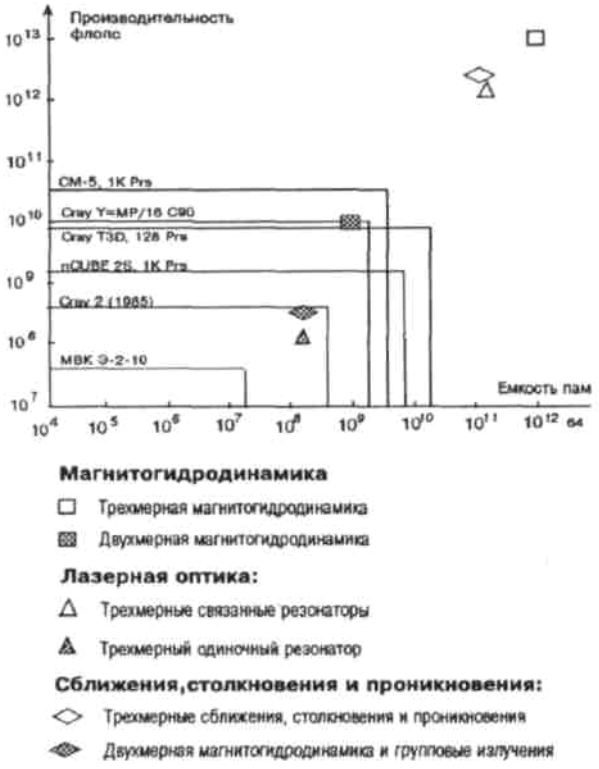


Рис.2

Отсюда можно сделать два вывода:

- суперЭВМ должна иметь эффективную и достаточно мощную жидкостную систему охлаждения;
- предельное быстродействие во многом зависит от качества логического элемента, его времени срабатывания tэ и выделяемой при этом энергии.

Безусловно, немалую роль при создании суперЭВМ играет показатель надежности элемента *X*. На практике зарубежных разработок этот параметр не ограничивает производительности, т.к. даже для схем большой интеграции величина λ не превышает 10^{-8}. Наша элементная база, в особенности на первом этапе производства, имеет $\lambda > 10^{-6}$, что заставляет вводить резервирование и

аппаратный контроль во все устройства машины, а последнее увеличивает объем устройств и потребляемую ими мощность.

Анализ показывает, что предельная производительность одного процессора, работающего по фон-Неймановскому принципу, вряд ли превзойдет 10^8оп/с на скалярных операциях [4]. Практика и анализ задач позволяет сделать вывод, что в одном процессоре конвейерного типа производительность на векторных операциях может быть поднята не более, чем на один порядок [5]. Увеличить производительность до 10^{14} оп/с, т.е. еще на пять порядков, можно только путем распараллеливания вычислительных процессов. Первым шагом в этом направлении являются многопроцессорные комплексы, как правило, состоящие из 4-16 процессоров, работающих на одну общую память.

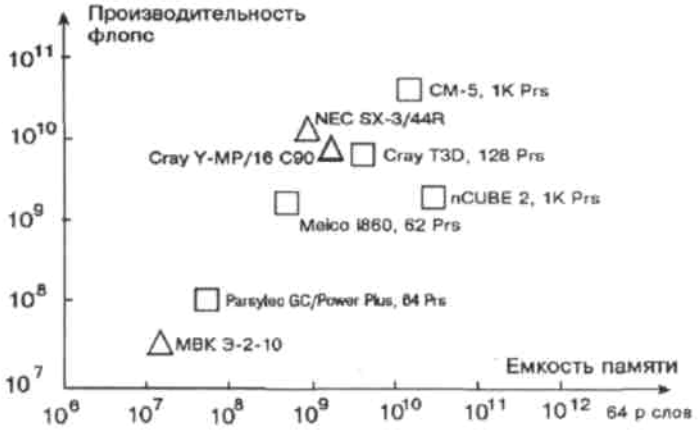


Рис.3.

Дальнейшее распараллеливание вычислительных процессов приводит к системам с массовым параллелизмом. Системы массового параллелизма насчитывают сотни, а иногда и тысячи параллельно работающих процессоров и строятся, как правило, на базе систем с распределенной по процессорам памятью.

Основное различие всех параллельных вычислительных систем сводится к способам коммутации процессоров. Так, многопроцессорные системы с распределяемой общей памятью имеют центральный коммутатор большой пропускной способности, обеспечивающий доступ каждого процессора к каждому модулю ОЗУ. Такие комплексы с небольшим количеством процессоров (1-16) можно считать достаточно универсальными. Увеличение процессоров в этой системе увеличивает время обращения к ОЗУ, что резко снижает быстродействие каждого процессора.

Системы с распределенной по процессорам памятью, как правило, имеют многоступенчатую систему коммутации.

Транспьютерная система коммутации, например, характерна тем, что каждый микропроцессор имеет четыре связи с соседними процессорами, как правило,

последовательного типа. Эти системы в большей степени ориентированы на решение двумерных задач.

Вычислительный комплекс Cray T3D имеет в каждом узле системы связи по трем взаимно перпендикулярным направлениям, причем каждое направление связи замкнуто в кольцо. Эта система максимально ориентирована на решение трехмерных задач.

СуперЭВМ СМ 5 и nCUBE 2-3 имеют межпроцессорные связи типа гиперкуб, причем, чем количество процессоров N больше, тем больше ранг коммутации d (согласно формуле $N=2^d$). Пользуясь таким представлением, можно коммутацию транспьютерных систем охарактеризовать как d=2, а коммутацию, принятую в T3D, как d=3 (Рис.4). Из рисунка видно, что передача данных между предельно отстоящими друг от друга процессорами в транспьютерной системе может быть осуществлена за $2\sqrt[2]{N}$ шагов, в системе T3D за $3/2\sqrt[3]{N}$ двойка в знаменателе появляется за счет замыкания ортогональных связей в кольцо), а в системе гиперкуба за d шагов. Все эти системы массового параллелизма являются в большей или меньшей степени специализированными системами, причем специализация их увеличивается с увеличением числа процессоров и степени ограничений, принятых в системе коммутации. Так, транспьютерная система массового параллелизма, имеющая большие ограничения связи между процессорами, является более специализированной, чем системы CM-5, nCUBE 2-3 и Cray T3D. В то же время, система Cray T3D, имеющая преимущества по решению трехмерных задач, является более специализированной системой, чем CM-5 и nCUBE 2-3. Этим и можно объяснить то обстоятельство, что Cray T3D используется только совместно с Cray Y-MP или Cray C90.

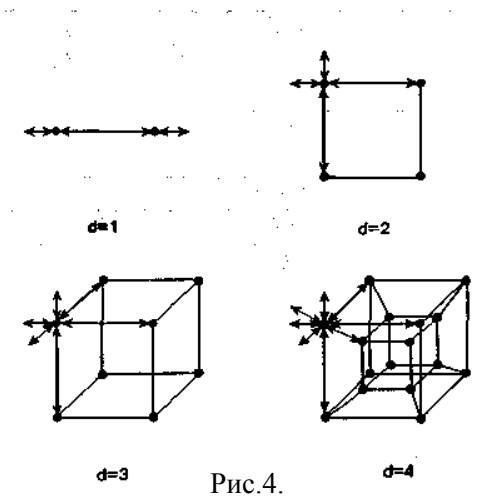


Рис.4.

Что же мешает многопроцессорным системам и системам MPP достичь предельной, наперед заданной производительности путем простого увеличения процессоров? Есть две основные причины, которые не позволяют это сделать в том случае, если вычислительный процесс реализуется по традиционному фон-Неймановскому принципу:

- первое - это большое время передачи данных между процессорами;
- второе - невозможность в статике при написании программ или при их трансляции эффективно использовать ресурсы системы: процессоры, память, каналы передачи информации и др.

Второе относится и к многопроцессорным вычислительным системам, работающим с распределяемой общей памятью.

Таблица 1 иллюстрирует загрузку процессоров в таких системах, которая резко падает от увеличения числа процессоров, даже на тестовых задачах. Человек

в статике не может загрузить такие системы, т.к. он не знает, сколько времени для того или иного вычислительного процесса будет занят процессор. Выход единственный - переход к распределению аппаратными средствами по свободным вычислительным ресурсам (процессорам, памяти, каналам передачи и др.), готовых к выполнению параллельных вычислительных процессов. Другими словами, достижение предельного параллелизма прохождения вычислительных процессов требует новых, нетрадиционных принципов их организации. Такие новые нетрадиционные вычислительные структуры MPP и у нас, и за рубежом успешно разрабатываются. Эти системы базируются на мощных системах коммутации, обладающих высоким темпом работы. Наверное, полупроводниковая вычислительная техника не сможет выдержать конкуренцию с оптоэлектронными коммутационными системами для новых вычислительных систем. Зарубежный анализ показывает (Рис.5), что в 1995-2000 г.г. оптоэлектронные коммутаторы вытеснят электронные [5].

No of Processors	**System**	**Peak speed gigaflops**	**Benchmark**	**Efficiency (code/peak)**
NAS Parallel Benchmark EP (size: 2^{28})				
16	Cray C90	16	8.3	50%
128	Intel iPSC/860	2.6	0.4	15%
65536	Thinking Machines CM-2	1620	2.4	12%
NAS Parallel Benchmark FFT (size: 256^2 x 128)				
16	Cray C90	16	4.5	30%
128	Intel iPSC/860	2.6	0.5	20%
65536	Thinking Vfchines CM-2	14	0.5	4%
NAS Parallel Benchmark CG (size 2 x 10^6)				
16	Cray C90	16	2.6	16%
128	Intel iPSC/860	2.6	0.07	3%
16384	Thinking Machines CM	35	0.7	3%

Source;NASA Ames Research Center,
Technical Report RNR-92 CC2
NAS = Numerical Aerodynamic Simulation

Таблица 1.

Итак, требуемая производительность 10^{14} оп/с в начале 2000-х годов, очевидно, будет достигнута. Спрашивается, необходимо ли нам заниматься суперЭВМ и можем ли мы создать конкурентоспособную суперЭВМ или нам

следует пойти по пути европейских стран и покупать суперЭВМ за рубежом. Есть несколько веских причин, говорящих о том, что необходимо создавать свои суперЭВМ.

Первое: даже если не принимать во внимание то, что военные объекты особой важности базируются и разрабатываются на основе суперЭВМ, решение перечисленных на Рис.1 проблем имеет определяющее значение для развития страны. Та страна, которая первой будет иметь возможнось решать эти задачи, всегда будет иметь преимущество в научно-техническом потенциале. Поэтому ни одна страна наиболее быстродействующую систему не продаст - мы будем иметь в лучшем случае системы "second hand" и решаемые задачи будут под контролем конкурентов.

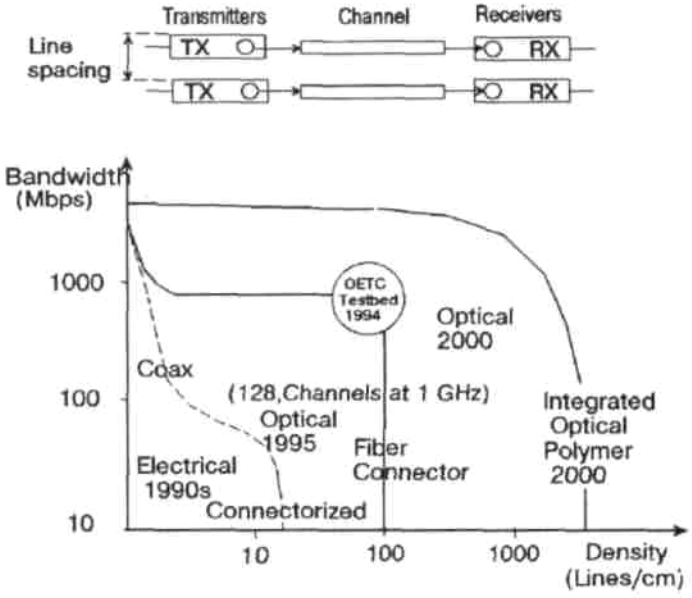


Рис. 5.

Второе: как показывает история становления вычислительных средств, суперЭВМ являются передовым фронтом развития всей вычислительной техники. Сохраняя разработки суперЭВМ, мы сохраняем передовые школы, определяющие развитие информатики в нашей стране.

Третье: в странах, где будут созданы самые высокопроизводительные вычислительные системы, будут и самые высокоинтеллектуальные талантливые научные кадры пользователей.

Четвертое: эксплуатация суперЭВМ, даже уровня "second hand", будет обходиться намного дороже, чем эксплуатация вычислительных средств собственной разработки.

Пятое: созданные в России суперЭВМ на современной зарубежной базе наверняка выиграют конкуренцию у зарубежных фирм как по архитектурным и схемотехническим решениям, включая системное матобеспечение, так и по

себестоимости. Эта наукоемкая продукция без сомнения найдет сбыт и позволит сохранить рентабельное производство вычислительной техники в целом.

Последнее заявление, безусловно, у многих вызовет по меньшей мере скепсис. Действительно, на вопрос: "Имели ли мы когда-нибудь суперЭВМ мирового уровня?" можно четко ответить: "Нет". В то время, когда мы заканчивали разработку, за рубежом всегда была суперЭВМ более быстродействующая, чем наша, построенная на элементной базе следующего поколения.

Но тем не менее, наши разработки по схемотехническим, архитектурным и конструктивным характеристикам зачастую превышали зарубежный уровень. Так, мы уже в 1970 году разработали идеологию многопроцессорной (10-процессорной) суперЭВМ, в то время как построения 4-процессорных комплексов в IBM кончались полной неудачей - резко падала общая производительность системы. Процессор МВК "Эльбрус-2" на элементах с задержкой τ =2 нс имел ту же производительность, что и компьютер типа Cyber на элементах с задержкой τ =1 нс. Разработанный в 1978 году векторный процессор, входящий в состав МВК "Эльбрус-2", имел большую команду, систему "разгона" и "торможения" и другие оригинальные решения, за счет которых на элементной базе МВК "Эльбрус-2" он достиг производительности 80 мегафлоп (той же, что и компьютер Cray на элементной базе с τ = 0,7). Созданный в ИТМ и ВТ векторный процессор МКП имеет производительность порядка 200 мегафлоп на элементной базе с τ =1,2 нс. Производительность достигается за счет оригинального схемотехнического решения - аппаратного распараллеливания скалярных и векторных операций. Векторный процессор ССБИС имел целый ряд оригинальных решений обработки данных на проходе и др. В настоящее время в РАН разработана и испытана на модели система массового параллелизма с автоматическим распределением ресурсов, имеющая целый ряд оригинальных решений. Система работает по новому не фон-Неймановскому принципу организации вычислительных процессов. Поэтому можно совершенно ответственно утверждать, что разработанная нашими специалистами суперЭВМ на современной элементной базе будет по многим параметрам превосходить зарубежные.

Безусловно, возникает вопрос, можно ли использовать зарубежную элементную базу и не станет ли она при определенных обстоятельствах камнем преткновения. Надо сказать, что в этом отношении суперЭВМ находится в лучшем положении, чем PC или рабочие станции. Последние изготавливаются и используются в больших количествах и обеспечение их зарубежной элементной базой довольно проблематично. В частности, если поставщик внесет изменения или сменит продукцию на новую, мы попадаем в полную зависимость от него.

С суперЭВМ дело обстоит гораздо проще. Как показывает зарубежный опыт (США), количество суперЭВМ в стране не превосходит двух десятков. Для России вполне достаточно иметь пять региональных вычислительных центров с развитой системой теледоступа, благодаря которой, как правило, почти каждый пользователь имеет доступ к суперЭВМ со своего рабочего места, и центры для исследовательских институтов различных направлений народного

хозяйства. На такое количество, и даже большее, можно закупить элементную базу на весь период эксплуатации.

Все это говорит о том, что в нашей стране необходимо создавать суперЭВМ как объект первостепенной важности. Это наукоемкое изделие требует в настоящее время разрешения целого ряда фундаментальных проблем, связанных с нетрадиционной организацией массовых параллельных вычислительных процессов. Многие из них нашли отражение в проекте ОСВМ РАН (посвященном созданию оптической сверхвысокопроизводительной машины), успешно защищенном в 1993 г.[6].

В заключение попытаемся определить, что же такое суперЭВМ и каковы ее основные характеристики, отличающие ее от других классов машин?

СуперЭВМ - это достаточно универсальные вычислительные средства, обладающие на данный момент времени максимальной производительностью.

Так, в 60-е годы к суперЭВМ относились машины производительностью более 1 млн. операций в секунду.

В 2000-х годах к суперЭВМ будут относится комплексы вычислительных средств производительностью выше $10^{13\text{-}14}$ операций в секунду. Вычислительные комплексы такой производительности, как правило, будут использовать жидкостное охлаждение, широкополосную оптоволоконную систему теледоступа, терабайтные хранилища информации с автоматизированным доступом к ней и развитую систему визуализации информации и процессов.

Каждый региональный центр на базе одной или нескольких суперЭВМ будет по каналам теледоступа обслуживать десятки тысяч пользователей, связанных с центром с помощью персональных компьютеров, рабочих станций и локальных линий связи.

Современные региональные вычислительные центры в США имеют общую производительность в десятки миллиардов операций в секунду и обслуживают тысячи пользователей.

Литература

1. Johnson G.M. Exploiting Parallelizm in Computational Science. North-Holland, Future Generation Computer Systems, 1989, 5, pp 319-337.
2. Van der Steen Aad J. Overview of Recent Supercomputers. The Netherlands, Academic Computing Centre Utrecht, 1994, 47 p.
3. Бурцев В.С. Система массового параллелизма с автоматическим распределением аппаратных средств суперЭВМ в процессе решения задачи. Юбилейный сборник трудов институтов Отделения информатики, вычислительной техники и автоматизации РАН, Москва, 1993, том II, стр 5-27.
4. Бурцев В.С. Тенденции развития высокопроизводительных систем и многопроцессорные вычислительные комплексы. Москва, 1977, 28 стр. (Препринт ИТМ и ВТ АН СССР)
5. Бурцев В.С., Кривошеев Е.А., Асриэли В.Д., Борисов П.В., Трегубое К.Я. Векторный процессор МВК "Эльбрус-2". СуперЭВМ, 1989, сборник научных трудов ОВМ АН СССР.
6. Отчет по ОСВМ. ВЦКП РАН, 1992.

Новые подходы к оценке качества вычислительных средств

В.С.Бурцев

Аннотация

В статье даются новые методы оценки структурных и архитектурных решений вычислительных средств. Проводится анализ современных высокопроизводительных многопроцессорных систем и многомашинных вычислительных комплексов. Делается вывод о том, что развитие вычислительных средств в настоящее время переживает определенный кризис, связанный с использованием в высокопараллельных структурах традиционного фон-Неймановского принципа организации вычислительного процесса. Делается вывод о необходимости перехода к созданию новых архитектур на суперпроцессорах, работающих на новых принципах и обладающих производительностью 10^{11}-10^{12} оп/с каждый.

Традиционно принято оценивать производительность вычислительных средств по скорости выполнения ими набора типовых тестовых задач. Это, безусловно, самый эффективный метод оценки производительности вычислительных комплексов, так как наряду с аппаратными средствами он в какой-то мере оценивает и эффективность системных программ. Однако, метод тестовых задач не учитывает времени затраченного программистом на подготовку этих задач. Нет так же ответа и на вопрос о возможности использования подобных вычислительных комплексов в открытых системах, так как этот метод не оценивает те сложности, которые возникнут в той или иной архитектуре при создании системного матобеспечения, работающего с открытыми системами. Метод не учитывает и квалификации программиста, реализующего адаптацию тестовой задачи к испытуемой аппаратуре. Поэтому результат тестирования зависит от многих факторов, вносящих в это процесс существенные искажения. Таким образом, этот метод не отвечает на целый ряд принципиальных вопросов, связанных с решением на современных комплексах сложных

проблемных задач, работающих над большим объемом взаимосвязанных данных. А именно ради решения таких задач создаются многомашинные и многопроцессорные средства.

Как известно [1], для решения целого ряда проблемных задач требуется производительность 10^{11}-10^{13} оп/с при работе с общей оперативной памятью объемом в 10^{9}-10^{10} слов. При современном состоянии развития вычислительной техники для решения этих задач потребуется сотни параллельно работающих процессоров.

Все это говорит о том, что должны быть созданы некоторые дополнительные оценки вычислительных средств, в какой-то мере отвечающие на эти вопросы. Нельзя сказать, что таких оценок по архитектуре и параметрам вычислительных средств не было вообще. Так пользователь и системный программист знали, что среди однопроцессорных машин лучшей является та, которая имеет большую производительность процессора, больший объем оперативной и внешней памяти, большую пропускную способность данных между оперативной и внешней памятью и т.д.

Однако, не существовало в какой-то мере систематизированной системы оценок даже для однопроцессорных систем, не говоря о многопроцессорных и многомашинных комплексах.

Попробуем на базе имеющихся в настоящее время общеизвестных суждений и эмпирических данных создать некую систему качественных сравнительных оценок вычислительных средств в интересах как системных программистов, так и программистов-пользователей. В настоящее время в промышленном выпуске существуют всего три архитектуры вычислительных средств: однопроцессорные машины, включая векторные конвейерные; многопроцессорные системы и многомашинные вычислительные комплексы [2]. В первую очередь дадим сравнительные оценки основных особенностей трех структур. Безусловно, с точки зрения простоты загрузки этих структур однопроцессорные машины имеют неоспоримое преимущество перед остальными. Действительно, и последовательные, и параллельные алгоритмы вычислительных процессов без особой изобретательности программиста достаточно эффективно выполняются на однопроцессорных машинах. Исключение составляют векторные конвейерные системы, для которых необходимо, чтобы количество параллельных вычислительных процессов выполняемой задачи, включая векторные, было больше коэффициента конвейеризации системы Ск, где Ск - глубина конвейеризации системы, измеряемая числом команд, одновременно находящихся в системе и выполняющихся на различных уровнях ее конвейера.

Для загрузки многопроцессорной системы к алгоритму необходимо предъявить определенные требования параллелизма вычислительного процесса, а именно: количество параллельно выполняемых процессов Р должно быть больше числа параллельно работающих процессоров системы на протяжении всего времени выполнения задачи. Необходимо отметить, что обмен данными между процессами в этих системах происходит посредством обычного обращения процессоров к общему ОЗУ, а общая синхронизация по данным реализуется на уровне специальных команд [3].

Многомашинные комплексы накладывают условия по параллелизму выполняемого алгоритма, аналогичные многопроцессорным, однако задачи межпроцессорного обмена (в данном случае межмашинные обмены) решаются гораздо сложнее - посредством взаимодействия операционных систем одной или нескольких машин. В то же время максимальное количество машин в таких комплексах может быть значительно большим, нежели количество процессоров в межпроцессорном комплексе. Последнее будет пояснено ниже.

Итак, из грубого анализа трех архитектур можно сделать следующий вывод. Если производительность однопроцессорного комплекса достаточна для решения задачи - необходимо использовать однопроцессорный. При одинаковом числе процессоров в многопроцессорной системе или машин в многомашинном комплексе, достаточном по производительности для выполнения задач, надо отдавать предпочтение многопроцессорной системе. И только в том случае, когда многопроцессорный комплекс при максимальном количестве процессоров не удовлетворяет по производительности, необходимо использовать многомашинный комплекс.

Это грубый анализ трех структур вычислительных средств. На самом деле на практике стоит вопрос не только определения глобальной архитектуры вычислительного комплекса, но и вопрос конкретного выбора вычислительных средств внутри одной и той же архитектуры. Несколько более глубокий сравнительный анализ отдельных машин, систем и комплексов должен идти по линии наиболее важных параметров каждого из них и их взаимной увязки.

Так, для однопроцессорных машин такими параметрами являются производительность процессора (Ппр), пропускная способность канала процессор-ОЗУ (Епр), объем ОЗУ (Q) и пропускная способность ОЗУ-внешняя память Е (последняя ограничивается в основном скоростью внешней памяти). Максимальная производительность однопроцессорной машины, как правило, ограничивается частотными характеристиками используемой элементной базы и соединений. Как правило, для всех систем выполняется следующее соотношение: Ппр=REпр, где R - процент обращений процессора к ОЗУ из всех обращений его за данными. Подразумевается, что большая часть обращений за данными идет в сверхоперативную память процессора (быстрые регистры, КЭШ и т.д.).Учитывая, что среднее число обращений за данными на одну операцию равно 2, а процент обращений к ОЗУ колеблется от 0,8 до 0,98 в зависимости от структуры задачи, можно с достаточной для наших рассуждений точностью считать Ппр=Епр, где Ппр имеет размерность Mflops, а Епр - MB/s. Поэтому, если мы не располагаем значениями Епр, можно определять их через Ппр. Производительность векторных конвейерных процессоров в дополнение к этому зависит от глубины конвейеризации, реализованной в процессоре. Опыт конструирования этих процессоров и анализ реальных задач показывает, что увеличение производительности процессора за счет конвейеризации не выше одного порядка.

Многопроцессорная система имеет следующие основные параметры: число процессоров N, производительность процессора Ппр, пропускная способность коммутатора между процессорами и ОЗУ Ек, пропускная способность ОЗУ-внешняя память Е. Во многих случаях Ек = NEпр = NППр

Максимальная производительность многопроцессорной системы ограничивается двумя факторами: пропускной способностью коммутатора между процессорами - ОЗУ (Ек) и требованием корректной работы припроцессорной КЭШ памяти. И та, и другая причины не позволяют строить многопроцессорные комплексы с большим количеством процессоров N. Аппаратная сложность коммутатора пропорциональна N^2. С увеличением сложности коммутатора растут временные задержки при обращении процессора к ОЗУ, что снижает скорость работы каждого процессора даже при наличии КЭШ при каждом процессоре. Практически увеличение числа процессоров выше 32 в одном коммутаторе вряд ли целесообразно. Исключение припроцессорной КЭШ памяти приведет к увеличению числа обращений процессоров к ОЗУ, что еще больше ограничит величину N. Обеспечение корректности работы КЭШ в многопроцессорной системе - сложная задача, и качество ее решения, в свою очередь влияет на производительность всей системы, особенно при больших N. Дело в том, что если два процессора поработали с общими данными, даже в том случае, когда синхронизация по данным для них была выполнена правильно, в КЭШ одного из процессоров могут сохраняться старые данные. При этом процессор, работая с обновленными другим процессором общими данными, может часть данных брать из ОЗУ (новые), а часть из КЭШ (старые). Корректность вычислительного процесса в этом случае будет нарушена. Простейшее решение проблемы корректности работы КЭШ в многопроцессорных системах состоит в том, что при каждой записи процессора в ОЗУ должно быть обеспечено стирание данных, записанных по этому адресу в КЭШ всех процессоров. Естественно, этот способ не позволяет увеличить число процессоров в многопроцессорных системах без значительного падения ее производительности. Наиболее эффективно эта задача была решена в МВК "Эльбрус", где число процессоров, одновременно работающих на общей памяти, было увеличено до 16 практически без потери производительности. Причем, способ обеспечения корректности КЭШ, реализованный в МВК "Эльбрус", практически инвариантен к числу процессоров системы - каждый процессор решает эту задачу самостоятельно без взаимодействия с соседними процессорами. Поэтому наиболее принципиальным препятствием в увеличении производительности многопроцессорных комплексов является коммутатор процессоры - ОЗУ.

Многомашинный комплекс должен решать достаточно сложную проблему обмена информацией между машинами, что осуществляется через взаимодействие операционных систем. Поэтому обмен информацией между машинами ведется, как правило, достаточно большими пакетами. Обмен малыми пакетами неэффективен из-за больших временных потерь, приведенных к одному слову. Может создаться впечатление, что проблема, подобная корректности КЭШ, в многомашинных комплексах отсутствует. На самом деле эта проблема при работе многих машин с общими данными переходит на уровень системных и пользовательских программ, что безусловно осложняет программирование задачи и увеличивает время ее выполнения. В то же время требования к временным параметрам системы коммутации машин становятся не такими жесткими, как в многопроцессорных комплексах, благодаря чему можно строить системы с большим числом машин N.

После такого общего анализа попробуем определить те параметры комплексов, которые накладывают определенные требования на системные программы и программы пользователей, имея в виду эффективное использование их аппаратных средств.

Начнем с однопроцессорной машины. Условием ее эффективного использования может служить критерий загрузки процессора. Для обеспечения загрузки однопроцессорной машины должно быть выполнено следующее неравенство на протяжении всего времени решения задачи:

$$Епр/Е \leq Ко \text{ при объеме памяти ОЗУ } Q, \qquad (1)$$

где Ко - средний процент переиспользования адресов ОЗУ на участке задачи объемом Q.

Действительно, если неравенство (1) не выполняется, то через определенное время производительность однопроцессорной машины будет определяться не величиной Епр, а Е, то есть пропускной способностью внешних устройств. Для того, чтобы этого не случилось, программист должен разбивать задачу на такие локальные части, для которых в объеме ОЗУ, равном Q, средний процент переиспользования адресов локальной части задачи, подкаченной в ОЗУ, превосходил величину Ко. Назовем величину Ко коэффициентом локализации. Естественно, чем больше объем памяти однопроцессорной машины Q, тем легче выполнить это требование. Поэтому условия выполнения неравенства (1) должны зависеть определенным образом от объема оперативной памяти. Примем некоторый объем ОЗУ Од, определенный практикой, за достаточный для локализации данных в ОЗУ. В этом случае для машин с памятью Q, меньшей Qд, усложняется проблема локализации данных, что может быть учтено соотношением (Qд/Q + 1), уточняющим неравенство (1):

$$Епр\ (Qд/Q + 1)\ /\ Е \leq К\ о \qquad (2)$$

В многопроцессорных системах условия обеспечения процессоров данными описывается подобным соотношением, в котором Епр заменяется величиной пропускной способности коммутатора процессоров - ОЗУ Ек (Ек = NEnp = NПпр). Предполагается, что все N процессоров имеют равную производительность Ппр и пропускную способность Епр. Тогда:

$$Кмп \geq Ек\ (Qд/Q + 1)/Е$$

Здесь Q - объем ОЗУ всего многопроцессорного комплекса, а Qд сохраняет прежнюю величину, так как мы фактически как бы увеличили производительность процессора однопроцессорной машины в N раз.

Однако для многопроцессорных систем появляется еще одно необходимое условие их эффективного использования. На протяжении времени выполнения всей задачи должно соблюдаться следующее неравенство:

$$Р \geq N,$$

где Р - число параллельно выполняемых процессов.

Необходимо отметить, что для мультипрограммного режима работы многопроцессорной системы это неравенство при достаточном количестве задач

выполняется без особых усилий. В то же время при решении больших задач, у которых N велико, оно не всегда может быть реализовано, что приводит к неэффективной загрузке системы. Поэтому, чем больше число процессоров в многопроцессорной системе, тем более специализированной она является [2].

Для многомашинных комплексов требования полной загрузки системы описываются несколько сложнее. Прежде всего, необходимо рассмотреть возможности выполнения межмашинного обмена с точки зрения эффективной загрузки узла комплекса. Каждый узел многомашинного комплекса можно представить в виде одного или нескольких процессоров, ОЗУ и коммутатора, связывающего этот узел с другими узлами комплекса (Рис.1).

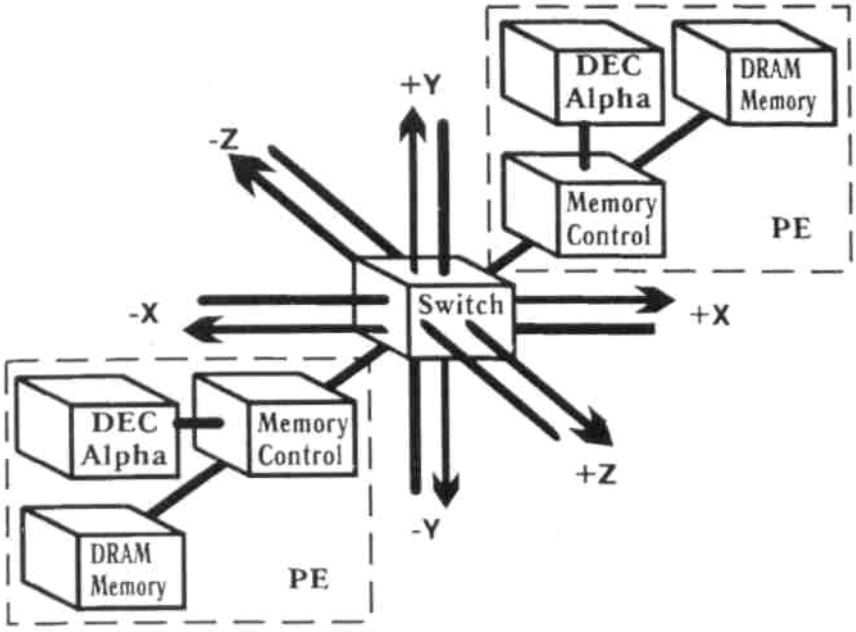


Рис.1. Схема узла Cray T3D.

PE - процессорный элемент, Memory Control - устройство управления памятью. DRAM Memory - оперативная память, Switch - коммутатор

В этом случае необходимое условие загрузки процессора или процессоров узла может быть описано неравенством:

$$Кму \geq NyEпр\ (Qд/Qy + 1)/Ey$$

где Еку - пропускная способность коммутатора узла со стороны процессоров, обычно равная NyEnp,

Ny - количество процессоров в узле,

Ey - общая пропускная способность коммутатора (узла) со стороны связи этого узла с другими узлами комплекса.

Необходимо отметить, что Qд имеет ту же величину, что и в предыдущем неравенстве, а Qy - объем памяти узла. Это обстоятельство сильно усложняет задачу удовлетворения этого неравенства. В дополнение к этому для относительной

оценки межмашинного обмена той или иной системы необходимо ввести коэффициент, отражающий топологию связей многомашинных комплексов.

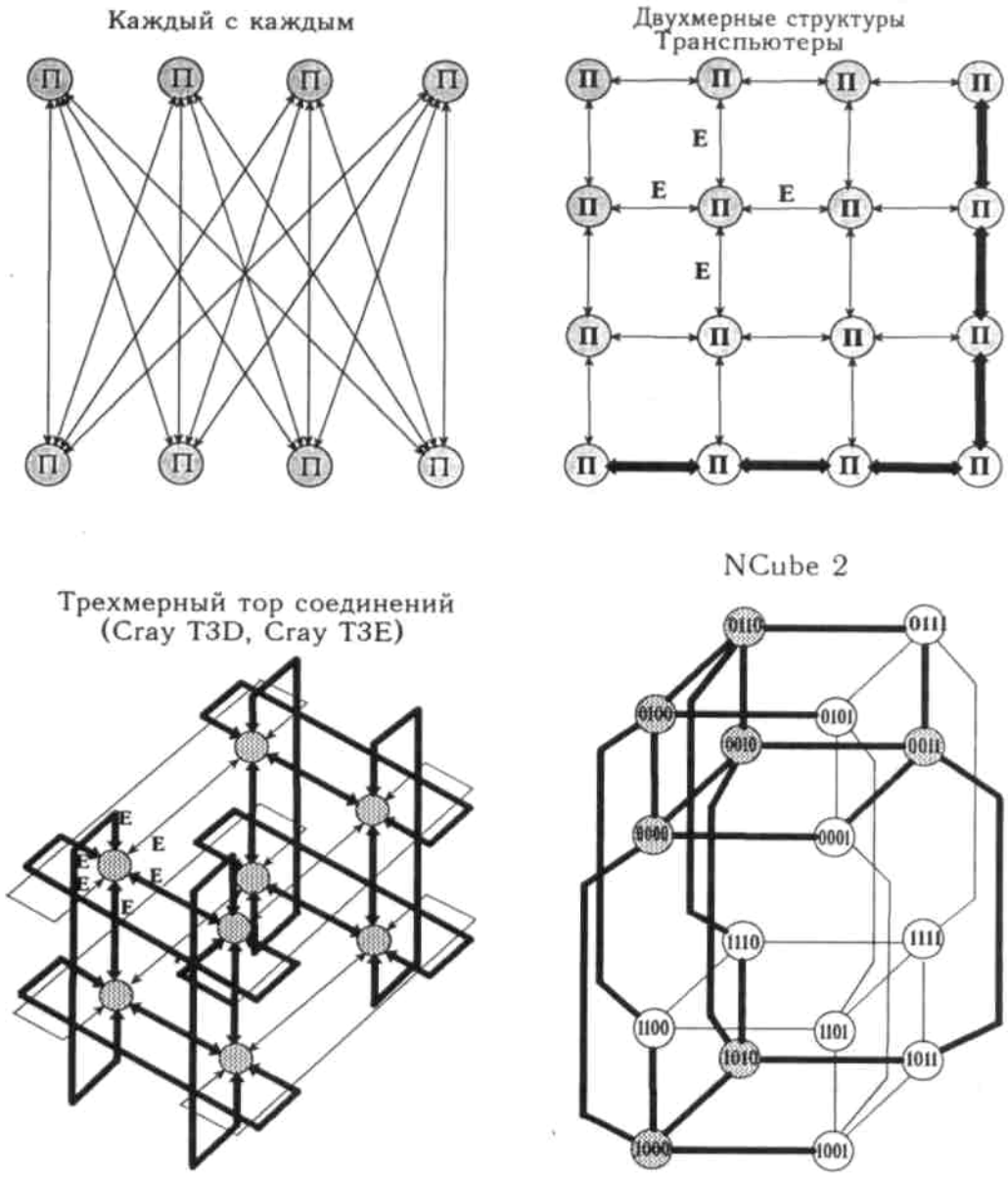


Рис.2. Схемы коммутации микропроцессоров в многомашинных комплексах.

В настоящее время реализованы следующие системы связей в многомашинных комплексах: "точка-точка", плоская матрица, трехмерная коммутация и система связей "гиперкуб" (Рис.2). Возможной характеристикой топологии связи может быть параметр, определяющий среднее число узлов передачи информации от узла к узлу. Так для системы "точка-точка" этот параметр b равен 1,

для транспьютерных связей b= $1/2\sqrt{N}$ (рассматриваются транспьютерные замкнутые системы), для трехмерной коммутации b = $3/4\sqrt[3]{N}$ (Cray T3D и T3E), для гиперкуба b= $1/2$ $1\log_2 N$ (Ncube 2). Для конкретных систем эти формулы могут несколько корректироваться, однако порядок зависимости параметра b от N останется тем же, и при сравнении многомашинных комплексов он должен быть учтен в определении величины Еу. В этом случае неравенство (2) уточниться следующим образом:

Кму ≥ NyEnp b (Qд/Qy + 1)/Еу (3)

Условие загрузки многомашинного комплекса при взаимодействии с внешней памятью может быть описано следующим соотношением:

Км ≥ Емк (Qд/Qм +1)/Е,

где Емк - пропускная способность всех коммутаторов системы, которая может определяться суммарной производительностью комплекса Пм = NПпр = NEпр, Qм - общий объем памяти комплекса, Qм = NQy. Условие загрузки многомашинного вычислительного комплекса при обеспечении требуемого уровня параллелизма вычислительного процесса совпадает с аналогичным условием для многопроцессорного комплекса:

P > N

Таблица 1. Соотношения для дополнительной оценки комплексов

Размер-ность величин	1 E - MB/s Q - GB	2 E - MB/s Q - GB $П_{ПР}$ - MFls	3 $П_{пр}$ - MFls Q - GB	4 $П_{ЭТ}$-MFls $П_{ПР}$-MFls	5
Однопро-цессорная машина	$K_0 \geq \geq \frac{E_{пр}}{E}\left(\frac{Q_д}{Q}+1\right)$		$K_{0П} = \frac{Q_П П_{пр}}{Q}+1$	Kпр= $=\frac{П_{эт}}{П_{пр}}$	$K_{ОР\ ЕЗ}$= $=K_О K_{ОП} K_{ПР}$
Многопро-цессорная система	$K_{МП} \geq \geq \frac{NE_{пр}}{E}\left(\frac{Q_д}{Q}+1\right)$		$K_{КПП} = \frac{Q_П NП_{пр}}{Q_м}+1$	Kпр= $=\frac{П_{эт}}{П_{пр}}$	$K_{МП\ рез}$= $=K_{МП} K_{МТП} K_{пр}$
Многома-шинный комплекс	$K_{МУ} \geq \geq \frac{bN_у E_{пр}}{E_у}\left(\frac{Q_д}{Q_у}+1\right)$	$K_М = \frac{NE_{пр}}{E_м}\left(\frac{Q_д}{Q_м}+1\right)$	$K_{МПП} = \frac{Q_П NП_{пр}}{Q_м}+1$	Kпр= $=\frac{П_{эт}}{П_{пр}}$	$K_{М\ рез}$= $=K_{МУ} K_М K_{МП} K_{пр}$

Важное значение при сравнении вычислительных комплексов имеет такой элементарный параметр, как относительное быстродействие процессора или микропроцессора системы Кпр = Пэт/П. Здесь Пэт - производительность микропроцессора, принятая за эталонную, которая выбирается как средняя

величина производительностей нескольких процессоров последнего выпуска Другим параметром является коэффициент достаточности памяти. Естественно, что чем меньше Кпр, тем ниже требования к распараллеливанию алгоритма для одной и той же задачи.

Практика использования вычислительных средств выявила следующую закономерность соотношения объема памяти и производительности системы $Q_п$, которая сохраняется на протяжении всего времени существования дискретных вычислительных средств - на каждый миллион операций в секунду приходится порядка 100 тысяч слов памяти (≈ 0.5 MB). Естественно, чем больше память, тем удобнее программировать задачу и повышать загрузку процессоров. Поэтому можно ввести специальный коэффициент, учитывающий достаточность памяти: в однопроцессорной машине Коп = 0,5 Ппр/Q + 1; в многопроцессорной системе Кмпп = 0,5NПпр/Q + 1; для многомашинного комплекса Кммп = 0,5Пм/Qм + 1. В этих соотношениях Ппр имеет размерность Mflops, а память Q и Qм - MB.

В Таблице 1 приведены все соотношения, по которым могут быть дополнительно оценены комплексы с точки зрения возможности их эффективного использования на разнообразных задачах пользователя. Другими словами, эти соотношения характеризуют сложность программирования на тех или иных системах, имея в виду, что всегда есть стремление достичь максимальной производительности вычислительных средств на решаемой задаче. Чем меньшие значения имеют правые части этих соотношений, тем легче выполнить условие полной загрузки системы. Произведение этих коэффициентов может дать качественную характеристику всего комплекса по эффективности его использования на различных классах задач. Для такой интегральной оценки необходимо правильно выбрать значения Qд и Пэт, так как они могут влиять на весовые характеристики этих коэффициентов. Анализ решения больших задач показывает, что для многих из них локализация данных во многих случаях может быть выполнена на объеме памяти Qд = 1 GB. Величина Пэт, как уже говорилось, может быть выбрана исходя из средней производительности процессора сегодняшнего дня lGFlops.

Приведенные оценки качества вычислительных комплексов указывают только возможные принципы нового подхода к анализу вычислительных средств при их выборе для использования в тех или иных сферах деятельности. Неполнота приведенных соотношений заключается прежде всего в том, что при определении пропускной способности канала процессор - ОЗУ не учитываются особенности построения сверхоперативной памяти (быстрые регистры, КЭШ, различные буферные устройства и т.д.). При определении пропускной способности между ОЗУ и внешней памятью не учитываются реальные возможности внешних устройств и телекоммуникационных систем, выходящих на вычислительные средства, и т.д. Однако, принципы развития и уточнения этих оценок достаточно ясны и могут быть без труда найдены для каждого конкретного случая.

Привлекательность приведенных оценок состоит в том, что все эти коэффициенты могут быть легко рассчитаны на основании рекламных данных, выдаваемых

фирмами. В качестве примера приведем анализ некоторых современных многопроцессорных систем и многомашинных комплексов.

Многопроцессорные системы имеют две различные структуры в зависимости от схемы построения коммутатора. Одна из них использует для коммутации данных общую шину, как, например, сервер фирмы DEC (Рис.3), другая -бесконфликтный коммутатор типа "каждый с каждым".

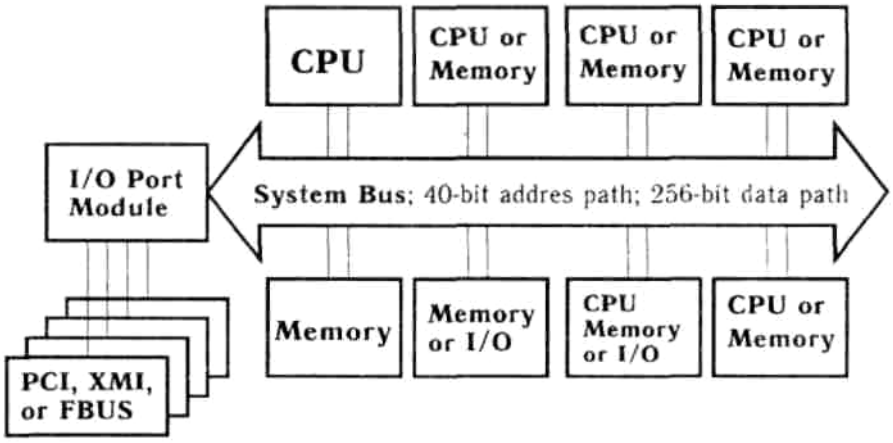


Рис.3. Структурная схема сервера AlphaServer 8400.

CPU - Процессор; Memory - Память; I/O Port Module - Модуль связи с устройствами ввода/вывода: System Bus - Общая системная шина; I/O - Ввод/вывод; PCI, XMI, FBUS - Интерфейсы ввода/ вывода.

Примером последней структуры может служить многопроцессорный комплекс SPP-1200 фирмы Convex при коммутации микропроцессоров внутри кластеров (Hypernode) (Рис.4).

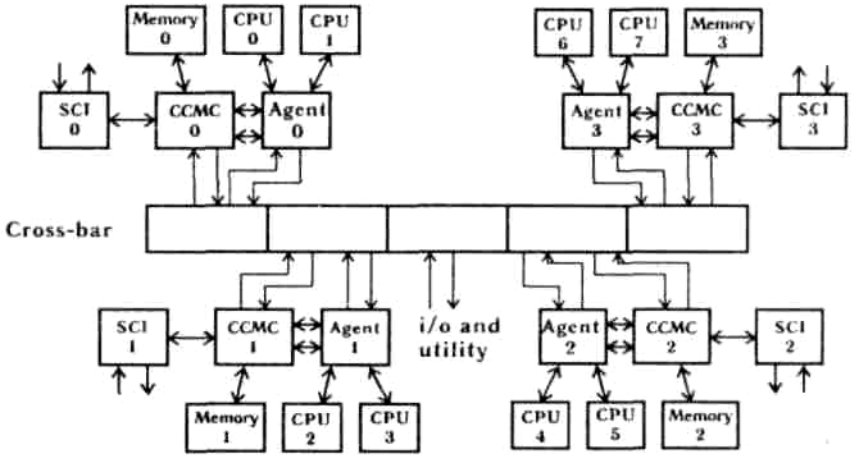


Рис.4. Структурная схема узла SPP 1200 фирмы Convex.

Cross-bar - коммутатор каждый с каждым (5x5); CPU - процессор; Memory - память; CCMC - локальный коммутатор процессор-память; SCL - локальный адаптер межузловой сети; Agent - адаптер; i/o and utility - ввод/вывод.

Многомашинные комплексы также отличаются структурой коммутатора: комплекс SP-2 имеет структуру коммутатора типа "точка-точка"; транспьютерные системы, выпускаемые НПО "Квант" (Россия) - структуру типа матрицы; в Cray T3D и T3E коммутатор организован по принципу трехмерного торроидального куба; а NCube 2 - по структуре гиперкуба. Особенности коммутации этих многомашинных комплексов в наших соотношениях учитываются коэффициентом b (Рис.2).

Результаты анализа многопроцессорных систем фирмы Convex (SPP-1200, SPP-2000), фирмы DEC (Server8400) и фирмы Cray (j916, C-90 и T-90) приведены в Таблице 2.

Таблица 2. Многопроцессорные системы (распределяемая память)

Фирма	Система	Базовый процессор			Параметры кластера					Коэффициент эффективности загрузки				R= $\frac{K_{рез}}{P}$	Примечания
		Наименование	f МГц	$П_{пр}$ GFls	N	$E_{пр}$ GB s	E GB s	P GFls	Q GB	$К_{мп}$	$К_{ммп}$	$К_{пр}$	$К_{рез}$		
CONVEX	SPP-1200	PA-7200	120	0.24	8	1	0.27	1.9	2	5.5	1.5	4	33	4	На аппаратно - программном уровне поддерживается объединение до 32 кластеров с коррекцией КЭШ
CONVEX	SPP-2000	PA-8000	180	0.7	16	7.6	1.72	11.2	4	4	2	1	8	0.7	
DEC	Server 8400	Alpha 21164	300	0.5	12	2.4	1.2	6.0	3	2	2	2	8	1.3	
Cray	Cray j916	Набор	100	0.2	16	26	1.6	3.2	4	16	1	5	80	26	
Cray	Cray C 90	Набор	240	1	16	245	13.6	16	8	18	1	1	18	1.1	
Cray	Cray T-90	Набор	470	2	32	960	54.4	64	16	18	1	0.5	9	0.13	

Аналогичные данные для многомашинных комплексов сведены в Таблице 3.

Таблица 3. Многомашинные комплексы (распределенная память)

Фирма	Система	Базовый процессор		Параметры кластера					Параметры комплекса						Коэффициент эффективности загрузки					R= $\frac{K_{рез}}{P}$
		Наименование	f МГц	$П_{пр}$ MFls	N_y	$E_{пр}$ MB s	E MB s	Q_y MB	в	N	$NE_{пр}$ GB s	E_M GB s	P GFls	M GB	$К_{му}$	$К_{ммп}$	$К_{пр}$	$К_м$	$К_{рез}$	
Cray	Cray T3D	Alpha 21064	150	150	2	37	76	64	3	1024	150	4.8	150	64	24	2	6	30	$8.6x10^3$	60
Cray	Cray T3E	Alpha 21164	300	600	1	1200	480	128	3	1024	600	64	600	128	22	4	1	9	$2.9x10^3$	4
IBM	SP2	Power-2	77	280	1	260	40	128	2	128	30	5	30	16	14	2	4	9	10^4	300
Квант	ИПМ Квант	Intel 860	40	80	1	80	4	16	3	128	10	0.05	10	2	3600	3	12	200	10^8	10^7

Из таблиц видно, что каждая фирма хорошо чувствует недостатки своей системы и при последующей реализации устраняет их, улучшая тем самым оценочные коэффициенты. Так, фирма Convex улучшила обмен к ОЗУ кластера с внешней памятью и каналами телекоммуникации и увеличила производительность процессора. Фирма Cray существенно увеличила объем ОЗУ в узле,

повысила его пропускную способность с другими узлами и увеличила общую пропускную способность с каналами связи и внешним полем.

Необходимо отметить также, что архитектура, предложенная НПО "Квант", очевидно, отражает скорее всего вынужденные решения, обусловленные чрезвычайно скромным финансированием этих работ. Несомненен также и тот факт, что многомашинные комплексы более специализированы, чем многопроцессорные системы с распределяемой памятью. При одних и тех же условиях программирование на многопроцессорных комплексах задач, обладающих достаточным параллелизмом алгоритмов, значительно сложнее, чем на однопроцессорных (с точки зрения достижения эффективной загрузки системы), однако значительно проще, чем в многомашинных системах с распределенной памятью.

Выводы

1. Предлагаемые дополнительные оценки различных архитектур вычислительных средств достаточно чутко реагируют на отклонения тех или иных систем и комплексов в сторону их специализации и дает возможность выявить диспропорцию в архитектуре вычислительных средств. Этим дополнительным тестом можно пользоваться как при выборе вычислительных средств, так и при их разработке.

2. Анализ существующих архитектур говорит о том, что многомашинные вычислительные комплексы не способны решить целый ряд проблемных задач, требующих производительности в 10^{12} и выше операций в секунду и оперирующих с большими объемами общих данных. Практически достаточная загрузка выходит за рамки человеческих возможностей. Многопроцессорные комплексы хотя и обеспечивают возможность обращения к общему ОЗУ, однако количество процессоров в этих комплексах имеет принципиальное физическое ограничение, не говоря уже о том, что загрузка таких комплексов с точки зрения синхронизации процессов, работающих с общими данными, представляется чрезвычайно сложным для программиста [2]. Все это говорит о том, что развитие вычислительных средств на базе фон-Неймановского принципа организации вычислительного процесса в настоящий момент претерпевает определенный кризис.

3. Приведенный анализ вычислительных средств свидетельствует о том, что единственным эффективным путем дальнейшего развития вычислительных средств в направлении повышения их производительности является путь существенного увеличения производительности процессора (Ппр). Именно за счет увеличения этого параметра Cray T90 имеет наилучшие показатели.

4. В то же самое время просматривается определенное насыщение в схемотехнических средствах проектирования микропроцессоров, состоящее в том, что при увеличении числа логических элементов (вентелей), в микропроцессорах не наблюдается пропорционального прироста его производительности или коэффициент прироста производительности существенно уменьшается [4].

Все это говорит за то, что традиционные принципы конструирования процессоров и микропроцессоров, основанные на фон-Неймановском принципе организации вычислительного процесса, себя изживают. Необходимы новые

принципы конструирования процессоров и микропроцессоров, которые позволили бы создавать суперпроцессоры, а затем и супермикропроцессоры производительностью в 10^{11}-10^{12} операций в секунду, исключающие их логическое насыщение. Нельзя сказать, что над созданием таких суперпроцессоров и супермикропроцессоров не было поставлено работ в России и за рубежом. В ИВВС РАН такие работы ведутся более пяти лет и в настоящее время с полной уверенностью можно сказать, что такие суперпроцессоры и супермикропроцессоры будут реализованы в недалеком будущем.

Литература

1. В.С.Бурцев. "О необходимости создания супер-ЭВМ в России" (в данной книге).
2. В.С.Бурцев. "Система массового параллелизма с автоматическим распределением аппаратных средств суперЭВМ в процессе решения задачи". Юбилейный сборник трудов институтов ОИВТА РАН. М. 1995, т.2, с. 5-27.
3. В.С.Бурцев. "Принципы построения многопроцессорных вычислительных комплексов "Эльбрус". М. 1977, ИТМиВТ, препринт N1.
4. Ю.Затуливетер. " Компьютерные архитектуры: неожиданные повороты". HARD V SOFT. М. 1996, N 2, с. 89-94.

Выбор новой системы организации выполнения высокопараллельных вычислительных процессов, примеры возможных архитектурных решений построения суперЭВМ.

В.С. Бурцев

Аннотация

Приводится анализ развития суперЭВМ. Обосновывается необходимость перехода от фон-Неймановского традиционного способа организации вычислительного процесса к новым принципам, основанным на синхронизации вычислительного процесса с помощью данных. Показываются преимущества этого способа в части автоматической загрузки аппаратных ресурсов процессора, особенно на задачах с высоким параллелизмом вычислительных процессов. Анализируется сложность аппаратной реализации этого принципа. Приводятся различные архитектурные построения комплекса и возможные схемотехнические решения построения отдельных устройств. Отмечается, что новые принципы организации вычислительного процесса хорошо адаптируются с оптическими принципами обработки информации. Производится оценка увеличения производительности как нового суперпроцессора, так и многопроцессорного комплекса, построенного на его базе.

1. Анализ архитектурных решений современных высокопроизводительных вычислительных комплексов

Построенная по принципу фон-Неймана однопроцессорная ЭВМ на скалярных операциях может достичь производительности 10^8 оп/с, а на векторных -10^9оп/с (Рис.1).

Возникло принципиальное физическое ограничение, связанное в основном со скоростью передачи информации от одного этапа логической обработки данных к другому, которая не может быть выше скорости света, в то время как сокращение расстояний между станциями логической обработки ограничено энергетическими соображениями. Выход из создавшейся ситуации в существенном распараллеливании вычислительного процесса.

Рис.1 Однопроцессорный комплекс.

ИУ - исполнительные устройства; М - модули оперативной и внешней памяти; E_1 - пропускная способность канала процессор - оперативная память; E_2 - пропускная способность каналов связи оперативной памяти с внешней памятью.

В настоящее время есть определенные успехи на уровне математических и численных методов решения задач в направлении распараллеливания вычислительных процессов. Аппаратные средства развивались в том же направлении. В структурах процессоров появилось несколько параллельно работающих исполнительных устройств, включая векторные. Широкое распространение получили многопроцессорные и многомашинные системы (Рис.2,3).

Все это дает возможность при передовой современной технологии достичь максимальной производительности комплекса 10^{10} - 10^{11} оп/с. Однако реальная его производительность, как правило, не превышает 10^{10} оп/с.

Основные причины, существенно сдерживающие возможности повышения реальной производительности вычислительных комплексов за счет распараллеливания вычислительных процессов, станут ясными после рассмотрения вопросов постановки сложных задач на вычислительные средства различной архитектуры.

С целью более четкого понимания излагаемого материала в рамках нашего контекста дадим определение исполнительного устройства, процессора и машины. Исполнительное устройство выполняет заданную обработку данных (операцию) над имеющимися данными (операндами), как правило, обеспечивая работу одной машинной команды. Процессор содержит одно или несколько

исполнительных устройств и обеспечивает выполнение одновременно одной или нескольких команд машины, организуя вычислительный процесс на этом участке вычислений. Процессор выполняет или участвует в выполнении почти всех команд машины.

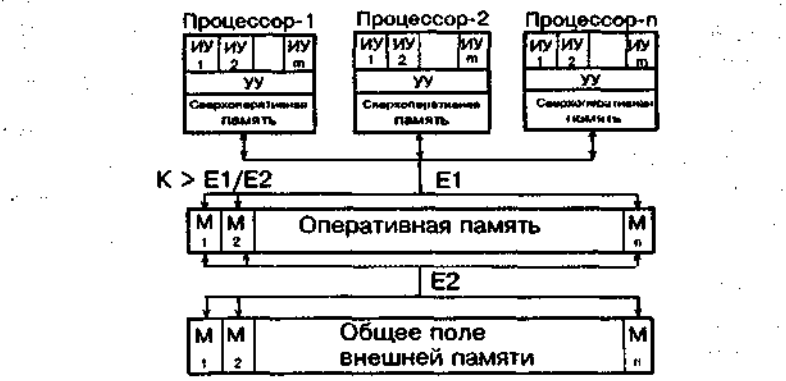


Рис.2 Многопроцессорный комплекс.

Рис.3 Многомашинный комплекс.

Характеристика процессора была бы не полной, если не назвать еще одно характерное для него свойство. Процессор без вмешательства человека обеспечивает предписанную алгоритмом выполняемой задачи последовательность выполнения команд. Это свойство чрезвычайно важно имея в виду то обстоятельство, что современные процессоры на аппаратном уровне осуществляют распараллеливание вычислительного процесса в локальной зоне выполняемых команд.

В настоящее время можно говорить о двух архитектурах суперЭВМ, отличающихся принципом распределения оперативной памяти по процессорам: жестко распределенная и распределяемая в процессе вычислений. Первые комплексы называют многомашинными, подчеркивая тем самым, что каждый процессор с принадлежащей ему оперативной памятью имеет свою операционную систему для распределения внутренних ресурсов и, в первую очередь, памяти. Вторые комплексы,

имеющие общую операционную систему для всех процессоров и памяти, называют многопроцессорными. Частным случаем этих двух архитектур являются однопроцессорные комплексы, иногда называемые машинами. Однопроцессорные комплексы могут включать в свою структуру векторные исполнительные устройства (например, CRAY 1). Таким образом, мы будем рассматривать три основные структуры вычислительных комплексов: однопроцессорные, многопроцессорные, многомашинные (Рис.1, 2, 3).

Мерой эффективности решения задачи на той или иной суперЭВМ может служить процент загрузки исполнительных устройств комплекса вычислительными процессами реализуемой задачи. Необходимо отметить, что постановка сложной задачи на вычислительные средства возможно более трудоемкая, чем разработка методов решения задачи.

Постановку сложной задачи на суперЭВМ можно разбить на следующие взаимосвязанные этапы работ:

1. Физическая постановка задачи.
2. Выбор математического метода решения задачи.
3. Разработка численных методов решения задачи.
4. Программирование.

Наилучший результат адаптации задачи к вычислительным средствам достигается в том случае, если работа проводится комплексно. Однако, на практике работы выполняются, как правило, последовательно, с вынужденными итерациями предыдущих этапов работ. Наиболее тяжелым является переход от третьего к четвертому этапу работ.

Дело в том, что решение той или иной задачи проблемного плана на этапах 1, 2, 3 сводится к отысканию алгоритмов ее решения для необходимой области данных с заданной точностью без учета тех ресурсов, на которых будет реализовываться этот алгоритм.

Алгоритм в данном случае описывает последовательность действий над данными без учета времен выполнения этих действий. Подразумевается, что действие выполняется мгновенно. В то же время, как только вводятся объекты, на которых производятся вычисления: исполнительные устройства, память нескольких уровней иерархии (быстрые регистры, сверхоперативная, оперативная, и внешняя память) и каналы, которые их соединяют - появляется необходимость учета времени занятости этих ресурсов.

Первым, кто сталкивается с необходимостью учета ресурсов вычислительных средств, а следовательно, и временными параметрами выполнения тех или иных операторов или подпрограмм, является программист, работающий на четвертом этапе разработки даже в том случае, если он работает на языке высокого уровня. Задача считается запрограммированной правильно тогда, когда соблюдается последовательность действий над данными, предписанная третьим этапом разработки задачи.

Проанализируем архитектуры различных суперЭВМ в отношении требований, предъявляемых к программам, разрабатываемым на четвертом этапе (Рис.1, 2, 3).

Сравним прежде всего различные архитектурные решения построения суперЭВМ с точки зрения возможности загрузки данными исполнительных устройств при одной и той же производительности комплекса.

1.1. Условия программирования однопроцессорных комплексов

Как указывалось в работе [4], существенным требованием к структуре программы, обеспечивающим загрузку исполнительных устройств однопроцессорного комплекса, является выполнение следующего неравенства:

$K > E_1/E_2$, где:

E_1 - средняя пропускная способность канала между процессорами и оперативной памятью;

E_2 - средняя пропускная способность канала между оперативной и внешней памятью;

K - коэффициент локализации данных программы в объеме оперативной памяти, равном Q, или средний коэффициент переиспользования адресов данных в объеме оперативной памяти, равном Q на интервале Δt (Рис.1).

На любом интервале времени At это неравенство должно удовлетворяться.

Необходимо отметить, что невыполнение этого неравенства приводит к резкому снижению производительности комплекса до величины, соответствующей пропускной способности канала E_2.

Принимая во внимание, что E_2 ограничено возможностями внешних устройств и в перспективе будет достигать 1-10 млн. слов в секунду (в настоящее время 0,1-1 млн. слов в секунду) при производительности процессора в 1-10 млрд. слов в секунду, коэффициент К измеряется сотнями единиц. Поэтому указанное неравенство обеспечивается, как правило, только при больших объемах оперативной памяти, что требует, даже на однопроцессорном комплексе, немалой изобретательности программиста, а иногда и пересмотра первого, второго и третьего этапов постановки задачи.

Отметим, что векторные конвейерные исполнительные устройства, входящие в однопроцессорные комплексы, повышают его производительность на порядок и не требуют от программиста выполнения каких-либо дополнительных условий, за исключением максимальной векторизации задачи [2].

1.2. Особенности программирования на многопроцессорных комплексах

В дополнение к изложенному выше условию разработки программы, в случае многопроцессорного комплекса, с целью загрузки всех процессоров пользователь должен создать такой алгоритм решения задачи, при котором на протяжении всего времени ее реализации существовало бы N или более независимых процессов (Рис.2). Это требование является необходимым, но не достаточным для эффективной загрузки комплекса. Для того, чтобы достаточно полно загрузить процессоры, программист должен с учетом временных интервалов выполнения отдельных процессов на процессорах оптимизировать график их работ и всего комплекса в интересах сокращения времени решения задачи.

Рассмотрим работу многопроцессорной архитектуры с точки зрения удовлетворения заданной последовательности выполнения операторов над данными при разработке программ на четвертом этапе. Как уже упоминалось, реальная реализация программы на вычислительных средствах, по сравнению с ее отображением на третьем этапе, состоит в том, что при составлении алгоритма на этом этапе предполагается мгновенное выполнение операторов и передача информации, в то время как на аппаратуре выполнение каждого оператора связано с занятием определенных ресурсов, таких, как канал связи, регистры, блоки памяти, исполнительные устройства и т.д. Использование ресурсов связано с временем их занятости, т.е. с временем выполнения операторов, которые в общем случае зависят от предистории занятости ресурсов. Так, даже в традиционной однопроцессорной ЭВМ это обстоятельство может привести к искажению вычислительного процесса. Например, вычисление выражения

d = b+c/k

может быть запрограммировано следующей последовательностью трехадресных команд:

1) y = +,b,c;
2) d = /,y,k.

После того, как заканчивается операция сложения в команде 1, может начаться выполнение команды 2 со считыванием ячейки y, причем, в зависимости от предистории использования ресурсов, считывание по адресу команды 2 может обогнать запись команды 1, в результате чего получится неверный результат.

В однопроцессорных системах контроль за правильностью выполнения операторов над данными возлагается на аппаратуру, т.к. имеются достаточно простые способы реализации этого требования.

Гораздо хуже обстоит дело с этой проблемой в многопроцессорных и многомашинных системах. Так, если в приведенном выше примере простые переменные b и c заменить на функции B и C, общие данные которых расположены в областях X и У, и предположить, что функции B и C могут параллельно выполняться на разных процессорах, вопрос о том, в какой временной последовательности функции B и C работали с общими данными, становится неопределенным. Синхронизация во времени этих двух вычислительных процессов осуществляется программистом при помощи специальных команд или операторов более высокого уровня. Обычно дело обстоит так, что, ужесточая синхронизацию процессов, программист, работающий на четвертом этапе, снижает параллелизм вычислений, а ослабляя синхронизацию процессов, увеличивает вероятность искажения вычислительного процесса. Кроме искажения счета в приведенном простом примере имеется возможность и полной блокировки вычислительного процесса. Так, если программа B, работая с данными X, обратилась к данным в области У в момент t_1, а программа C, работая с данными У с момента t_2, обратилась к данным X в момент t_3, где $t_3>t_1>t_2$, то программа B ждет данных X, а программа C - данных У, т.е. вычислительный процесс блокирован. В случае $t_2>t_1$ процесс прошел бы нормально.

Для обеспечения однозначности вычислительного процесса необходимо, чтобы программа, изменяя общие данные, блокировала бы обращения к ним других процессов на время своей работы с ними.

Как уже было сказано, еще хуже обстоит дело с требованием наиболее полной загрузки процессоров. Для того, чтобы произвести их загрузку, программист должен располагать временной диаграммой работы всех ресурсов комплекса на различных участках программы, что, естественно, невозможно сделать в случае большого количества процессоров.

Естественно, что чем больше параллелизм выполняемой программы, тем сложнее избежать искажений вычислительного процесса, связанного с внесением временных факторов исполнения отдельных его частей на вычислительных средствах. Как правило, синхронизация вычислительных процессов снижает их возможный параллелизм. В многопроцессорных комплексах, очевидно, справедлив тот факт, что с увеличением числа процессоров комплекс теряет свою универсальность. Не случайно число процессоров в многопроцессорных суперЭВМ, как правило, не превышает 16. Таким образом, многопроцессорный комплекс, построенный на традиционных принципах с широким использованием векторных процессоров, может увеличить общую производительность вычислительных средств не более чем на два порядка (один порядок за счет векторного конвейера, один за счет многопроцессорности).

1.3. Особенности программирования многомашинных комплексов

В многомашинных комплексах требования постановки задачи несколько отличаются. Дело в том, что взаимодействие процессоров в многопроцессорном комплексе можно реализовать практически мгновенно - процессоры между собой могут взаимодействовать через канал чрезвычайно большой пропускной способности (процессор - оперативная память), где для временной синхронизации обмена используется оперативная память комплекса. В многомашинном комплексе вся задача размещается по N отдельным устройствам оперативной памяти каждого процессора (Qэвм), а взаимодействие между данными осуществляется по межмашинным связям. Даже в том случае, если бы удалось реализовать межмашинные связи каждой ЭВМ с каждой, задачу полного обмена информацией между ЭВМ можно было бы осуществлять в N раз медленнее, чем этот же обмен между оперативной памятью и процессором, суммарная производительность которого равна производительности всех ЭВМ многомашинного комплекса. Считается, что пропускная способность канала одной ЭВМ в словах не может быть больше производительности ее процессора (Рис.3).

Различные способы организации локальной сети ЭВМ в многомашинном комплексе, такие, как общая шина, кольцо или гиперкуб и т.д., облегчают техническую реализацию межмашинных связей, ухудшая в той или иной мере возможности передачи информации по сравнению с соединением каждой ЭВМ с каждой. С учетом последнего можно сформулировать минимальные требования к задаче пользователя для постановки ее на многомашинный комплекс.

Задача на всем протяжении ее выполнения должна быть разбита на N параллельно выполняемых участков с объемом данных по памяти, не превышающим Qэвм, и степенью локализации Кэвм > N.

Естественно, что для большинства вышеперечисленных задач при достаточно большом числе N (N > 1000), эти требования практически нереализуемы.

С точки зрения обеспечения требуемой последовательности работы операторов над данными и сложностей, возникающих из-за блокирования вычислительного процесса, в многомашинных комплексах проблемы те же, что и в многопроцессорных.

Таким образом, построение суперЭВМ на базе малых ЭВМ производительностью в 5-7 млн. оп/с не может рассматриваться в качестве перспективного направления создания универсальных вычислительных комплексов производительностью в миллиарды и более операций в секунду.

Однако, было бы неверным говорить, что "Connection machine", имеющая в своем составе более 64 тыс. микроЭВМ, nCUBE-2 или "CRAY-4", не имеет права на существование в области решения больших задач. Можно сказать только, что с увеличением числа ЭВМ или процессоров в комплексе, супер-ЭВМ становится все более специализированным комплексом, т.е. эффективно решает все меньший круг задач.

Таким образом, препятствием к использованию систем с высоким параллелизмом вычислительных средств является нерешенная проблема, связанная с невозможностью эффективной реализации на этих средствах программы, несмотря на то, что она имеет высокопараллельный алгоритм, описанный на уровне численных методов.

Дело в том, что программа предполагает для выполнения параллельных участков программ определенные ресурсы вычислительных средств (процессор, память, каналы, диски и т.д.). Как только появляется работа с ресурсами, возникает вопрос, какое время эти ресурсы используются. Программист должен точно знать время выполнения отдельных параллельных процессов, которое зачастую зависит от самих данных, и стараться так распределить ресурсы вычислительных средств, чтобы простой оборудования был минимальным. При решении сложных задач, человек не в состоянии корректно решить эту задачу. Такую задачу может решить только аппаратура, имеющая информацию о динамике вычислительных процессов в каждый момент времени. Это обстоятельство вынуждает перейти к новой, не фон-Неймановской архитектуре суперЭВМ [3].

2. Особенности организации вычислительного процесса в машинах, работающих по принципу потока данных

Отход от традиционных принципов обработки информации и переход к организации вычислительного процесса по мере готовности данных позволяет решить вопрос максимального распараллеливания вычислительного процесса во времени. Степень параллельности в принципе ограничена лишь последовательностью вычислений, заданной только алгоритмом обработки данных на этапах постановки задачи 1, 2, 3. Этим обстоятельством можно объяснить такой

интерес к архитектурам машин, у которых последовательность операций во времени определена готовностью данных. Основные принципы работы таких суперЭВМ сводятся к следующему.

1. Алгоритм решения задачи, описанный на третьем этапе, представляется графом вычислительного процесса без переиспользования цепей графа для различных данных (Рис.4.).

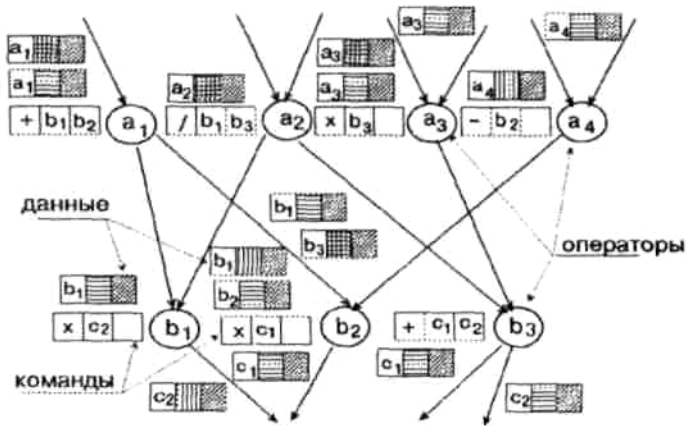


Рис.4 Граф вычислительного процесса.

Граф состоит из операторов над данными и указателей, по которым перемещаются данные к следующему оператору. Таким образом фактически определяется последовательность обработки данных.

2. Обработка данных в соответствии с последовательностью, определяемой графом, ведется по мере готовности операторов к выполнению соответствующей обработки данных, т.е. наличия необходимых данных на входе оператора. Считаем, что количество исполнительных устройств не ограничено, а время передачи данных между операторами $t << t_{оп}$, где $t_{оп}$ - минимальное время обработки данных оператором.

Очевидно, что при такой организации вычислительного процесса исключена вероятность искажения вычислительного процесса и его самоблокирования при правильно составленной программе, т.к. оператор обрабатывает только данные, поступившие к нему на вход. Отсутствуют глобальные данные и так называемые "побочные эффекты". Каждый оператор является функцией, поставляющей значение. При параллельно выполняемой обработке данных исключается возможность использования устаревших данных, т.к. нет повторного использования цепей графа, а, следовательно, данные к поименованному оператору обращаются только один раз, что эквивалентно принципу, заложенному в языках одноразового присвоения имен.

В то же время весь параллелизм вычислительных процессов, заложенный на трех первых этапах разработки алгоритма задачи, эффективно реализуется, т.к. все операторы, готовые к выполнению, работают параллельно.

2.1. Сложности аппаратной организации

Безусловно, на пути создания суперЭВМ, работающей по новым принципам, лежит множество на первый взгляд кажущихся неразрешимыми проблем. Основные из них следующие:

1. Требуется значительная избыточность объема памяти за счет хранения возможных ветвей графа, вычислительный процесс по которым не пройдет при реализуемых величинах данных.
2. Ввиду того, что имена операторов не повторяются, каждому данному и оператору должен быть придан уникальный номер (адрес). В этом случае поле адреса в команде может превысить поле данных, что по меньшей мере удвоит необходимый объем оперативной памяти.
3. Необходимое количество итераций вычислительного процесса заранее, как правило, не известно и определяется в процессе счета по данным. Поэтому граф необходимо "разворачивать" на максимум итераций, который неизвестен.
4. Оперативная память для решения поставленных задач должна иметь время доступа менее 1 нс и период темпа приема и выдачи информации менее 10 пс, обладая объемом, намного большим чем 10^{15} слов.
5. Слишком малый коэффициент использования оборудования. Так, для максимального распараллеливания умножения трехмерных матриц размерностью 100x100x100 требуется 10^6 умножающих устройств, из которых в среднем будет загружено 100. Еще меньшим коэффициентом загрузки будет обладать оперативная память при большом числе вариантов прохождения программы и неопределенности числа итераций по подпрограммам.
6. Чрезвычайно неудобно работать в этой структуре с многократным использованием данного (типа константы).
7. Принципиально неразрешимым кажется вопрос работы со структурами данных при одновременном обращении к ним из разных мест графа.
8. Чрезвычайно не эффективна, по сравнению с традиционными машинами, работа на участках программ с сильно связанными данными.

Особенности работы тех или иных машин потока данных связаны с практической реализацией вышеперечисленных проблем. Наибольший интерес представляют следующие разработки: работы Массачусетского университета США, Манчестерского университета Англии, проекты Sigma 1 - Sigma 4 лаборатории электроники, разрабатываемый в настоящее время проект EM-5 в Японии, проекты Monsoon и Epsilon в США, проект CSRO-2 в Австралии и др. Как правило, разрабатываются компромиссные варианты, использующие как новые, так и традиционные методы организации вычислительных процессов.

Типовой машиной потока данных, работающей на динамических принципах, можно считать ЭВМ, реализованную в Манчестерском университете (Рис.5).

Эта ЭВМ представляет собой вычислительное кольцо, состоящее из блока памяти ожидания совпадений для спаривания токенов, относящихся к одной команде, объединяющего устройства и устройства переполнений. Спаренные токены посылаются на устройство подготовки, которое добавляет к ним код исполняемой команды и адреса, по которым необходимо отсылать результаты.

Готовые пакеты поступают для исполнения на двадцать одинаковых универсальных арифметикологических исполнительных устройств. Результаты выполнения команды передаются на коммутатор, который согласно адресу назначения отсылает часть токенов в глобальную коммуникационную сеть, а остальные попадают во входную очередь токенов, предназначенную для сглаживания темпов генерации и их исполнения в следующем цикле работы машины.

В рамках работ по "Оптической сверхвысокопроизводительной вычислительной машине" ОСВМ [5], проводившихся ВЦКП РАН (ныне ИВВС РАН) совместно с ведущими физическими институтами РАН, создана новая нетрадиционная высокопараллельная архитектура вычислительных средств супер-ЭВМ на базе сетевой системы с потоковым принципом обработки данных для решения сложных задач вычислительного класса.

Эта суперЭВМ позволяет автоматически решать вопросы распределения ресурсов вычислительных средств с дискретностью до операции, освобождая человека от непосильной для него работы и тем самым существенно расширяя пределы производительности вычислительных средств.

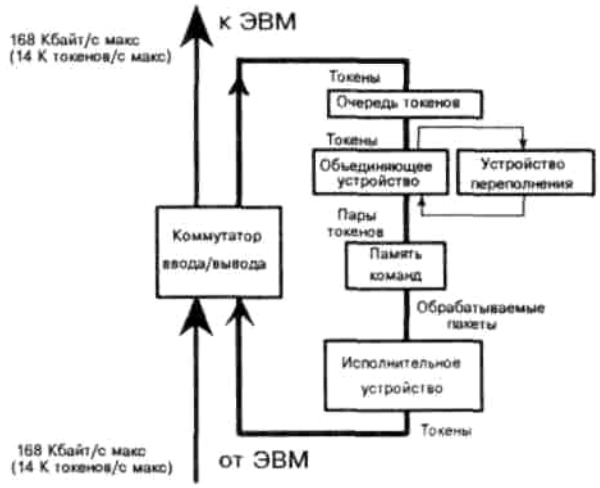


Рис.5 Манчестерская машина

Проект имеет целый ряд оригинальных структурных и схемотехнических решений, отличных от зарубежных аналогов и позволяющих переходить к проектированию этой системы для практического использования.

3. Принципиальные особенности реализации машины потока данных в проекте ОСВМ

Остановимся на принципиальных особенностях проекта или на том, как решаются в нем основные проблемы, возникающие при проектировании суперЭВМ потока данных.

Особенности реализации обусловлены следующей целью проекта:

а) достижение предельного быстродействия вычислительных средств за счет массового параллелизма выполнения программ с дискретностью до операции;

б) максимальное исключение человека из распределения ресурсов вычислительных средств.

3.1. Память совпадения на аппаратной основе

В связи с необходимостью получения максимальной производительности системы, исключена возможность построения памяти объединения на базе обычной прямоадресуемой памяти с использованием программно-аппаратных средств.

Фактически, память объединения может быть реализована на базе ассоциативной памяти выборки информации по ключам. Как показано в [3], предельная производительность этой памяти обратно пропорциональна ее объему.

Наилучшим образом эта память может быть реализована на оптических принципах. Для такой памяти при объеме ее ассоциативной части в 10^9 ключей можно достичь максимальной производительности порядка 10^7 операций. Необходимо отметить, что структура потока данных чрезвычайно эффективно работает при выполнении векторных и матричных операций. Так, умножение матриц на такой структуре будет выполняться вдвое эффективнее, чем на сисcталическом процессоре. Поэтому, если рассматривать память совпадения как единственную память системы, требования к ее объему могут измеряться миллиардами слов.

Было решено использовать несколько способов сокращения объема ассоциативной памяти и повышения ее производительности. Существенное сокращение объема ассоциативной памяти может дать введение специальной векторной памяти значительного объема. Однако, в отличие от проекта M1T, предполагается память структур и векторов малых размеров оставить в ассоциативной памяти и реализовать их работу за счет специальных операций исполнительных устройств и самой ассоциативной памяти.

Векторную память разместить в специальном векторном исполнительном устройстве и выполнить ее с дискретом в 100 слов с прямым доступом по адресам. Векторная память в скалярной машине потока данных представляется памятью массивов (размером не более 10000 слов каждый), имеющих свое виртуальное имя. Каждый такой массив в скалярной машине имеет статус операнда, из которого стандартным способом формируется токен. По готовности таких токенов формируется пакет и запускается операция векторного испытательного устройства (Рис.6).

Для увеличения производительности работы ассоциативной памяти при сохранении достаточно больших объемов предложены схемотехнические решения, позволяющие разбивать общую память на модули. При этом пропускная способность памяти увеличивается как за счет увеличения количества модулей, так и за счет увеличения скорости работы самого модуля. При той же общей мощности памяти производительность возрастает в N раз, где N - число модулей памяти. С этой целью в структуру машины введен коммутатор $К_2$.

Коммутатор K_1 производит распределение готовых пакетов токенов по свободным исполнительным устройствам, включая векторные. Готовые пакеты, не находящие свободных исполнительных устройств, сбрасываются в буферную память (БП). Таким образом, в структуру машины введена специальная буферная память прямого доступа достаточно больших размеров, которая хранит избыточную, готовую к выполнению информацию (пакеты токенов) в том случае, когда возникают так называемые информационные "взрывы". Несколько слов о происхождении "взрывов". Из графа видно, что есть операции, которые из двух операндов формируют один операнд, но есть и такие, которые из одного операнда формируют два. Если бы существовали только операции первого вида, то граф свернулся бы к одному узлу. Существование операций только второго вида существенно расширило бы количество параллельно выполняемых операций в принципе до бесконечности.

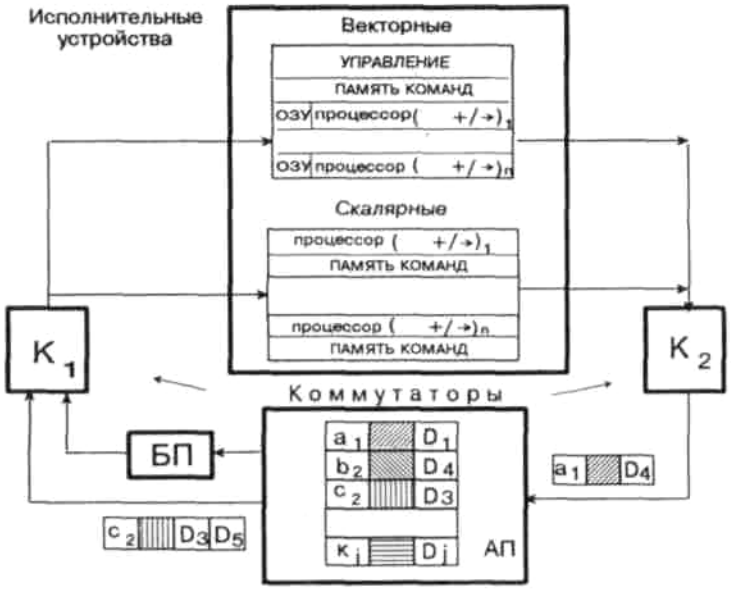


Рис.6 Блок-схема машины.

При существенном расширении графа (Рис.4.) готовые к выполнению пакеты складываются в буферную память по принципу "первый пришел-первый выйдет". Этот принцип необходимо выдержать при работе всего кольца машины потока данных с целью уменьшения необходимого объема ассоциативной памяти. Другим способом нивелирования "взрывов" является торможение его развития. Дело в том, что мы можем иметь информацию о загрузке ассоциативной памяти в целом. Ее перегрузка может возникать из-за большого параллелизма вычислительного процесса, который порождается совершенно определенными операциями, вводящими новые индексы, итерации и активации и др. Поэтому, чтобы предотвратить переполнение АП, можно, не нарушая па-

раллелизма выполняемого вычислительного процесса, отложить в буферную память операции, ведущие к переполнению памяти на этот момент.

Несмотря на все эти меры, ассоциативная память является единственным звеном, ограничивающим скорость работы потоковой машины при работе над задачами, обладающими большим параллелизмом вычислительного процесса.

Действительно, как бы хорошо ни была выбрана функция распределения информации по модулям ассоциативной памяти, всегда будут иметь место конфликтные ситуации при обращении к одному и тому же модулю с нескольких направлений, в то время как увеличение числа модулей при том же общем ее объеме снижает объем памяти каждого модуля и увеличивает вероятность его переполнения. И в том, и в другом случае происходит снижение производительности памяти в целом **[4].**

Кардинально эта проблема может быть решена только с использованием ассоциативной памяти на оптических принципах, поскольку оптика позволяет осуществлять одновременный поиск по нескольким ключам внутри одного модуля, что исключает замедление работы памяти при конфликтных ситуациях.

Несмотря на то, что предложенные новые принципиальные решения направлены к снижению требуемого объема ассоциативной памяти, необходимо отметить следующее:

- объем ассоциативной памяти является тем ресурсом, который необходимо учитывать системному программисту;
- объем ассоциативной памяти во многом определяет диапазон решаемых задач и реальный параллелизм их исполнения.

3.2. Вопрос неопределенного количества итераций, активаций и циклов. Многократное использование программного кода и константы

В проекте широко используется метод "раскраски" данных. В составе токена наряду с полем, определяющим номер команды, включены поля номера активации, номера итерации, номера индексации. Проработана аппаратная поддержка работы с этими полями [4]. Одновременно с этим предложен и программный вариант реализации этих функций с поддержкой таких операций, как вход в процедуру и выход из процедуры на уровне макрокоманд. Последний в настоящее время, очевидно, будет более предпочтительным в виду того, что не имеется опыта эксплуатации таких машин, и не ясно, какие из конструкций необходимо реализовывать в аппаратуре, а какие лучше оставить на программы.

Введение гибкой и мощной системы индексации, активации и итерации в программирование дает возможность, не теряя параллелизма, использовать неизменяемый код программы, инвариантный для любого места вычислительного процесса. Однако встает вопрос реализации командной памяти с высокой пропускной способностью по считыванию команд. В окончательном варианте проекта вопрос высокой пропускной способности решается дублированием командной памяти в соответствии с числом исполнительных устройств (Рис.6). Другим вариантом реализации этой памяти может быть оптическая память с относительно большим временем записи и малым временем считывания.

Существенные сложности при программировании в системе потока данных возникают в работе с константами, в особенности с так называемыми "константами на время действия подпрограммы или процедуры". Вопрос с константами решен при помощи распределенной памяти программ, в которую все употребляемые в задаче константы записываются вместе с кодом команды и сохраняются там неизменными в процессе всего времени решения задачи. Для удобства организации локальной константы в системе команд предусмотрен специальный индексный механизм и непосредственная команда ассоциативной памяти считывания без стирания информации.

В проекте ОСВМ оригинальными архитектурными средствами также решен вопрос устранения засорения памяти, эффективной работы с сильно связанными данными и ряд других проблем, принципиальных для этой структуры.

4. Особенности программирования и оценка производительности

Основным преимуществом машин потока данных является предельно возможная бесконфликтная автоматическая загрузка вычислительных средств с дискретностью до одной операции. Безусловно, требует определения величина предельной возможности, поэтому остановимся на этом подробнее.

Если отвлечься от ресурсов вычислительных средств и, следовательно, от временных факторов протекания вычислительного процесса, то любой алгоритм вычислительного процесса можно разбить на группы независимых операций, последовательно обрабатываемых машиной потока данных - это как бы горизонтальный срез графа вычислений.

Количество операций или узлов графа, находящихся в одной такой группе, определяет параллелизм вычислительного процесса на данном этапе или шаге вычисления.

Мы уже говорили, что необходимым условием полной загрузки вычислительных комплексов (многомашинных или многопроцессорных) должно быть выполнение следующего неравенства на каждом шаге вычисления:

$N < n$,

где N - количество процессоров или машин, а n - величина параллелизма вычислительного процесса.

Необходимо отметить, что это только нижняя оценка требуемого неравенства. Это неравенство должно быть уточнено в той части, что N необходимо умножить на величину C, характеризующую максимально возможное количество операций в каждом процессоре или машине, которое должно в них одновременно обрабатываться для обеспечения полной загрузки вычислительного комплекса. Фактически должна быть учтена глубина конвейеризации по каждому параллельно работающему процессору или машине.

В этом случае неравенство примет следующий вид:

$N \times C < n$.

Поэтому, предельные возможности по загрузке разрабатываемой вычислительной системы можно оценить следующими соотношениями:

если $NC < n$, то полная загрузка;

если $NC > n$, то загрузка будет пропорциональна соотношению n / NC.

Необходимо уточнить, что такое NC, например, для структуры, показанной на Рис.6.

$$NC = (N_{ПР}C_{ПР} + N_{к1}C_{кl} + N_{К2}C_{К2} + N_{АП}C_{АП})N_К,$$

где $N_{ПР}$, $N_{АП}$, $N_{К1}$, $N_{К2}$ - количество параллельно работающих устройств процессоров, модулей памяти и коммутаторов в одном канале; Спр, Сдп, $C_{к1}$,$C_{к2}$ - соответствующие величины глубины их конвейеризации.

С целью упрощения задачи будем считать, что пропускные способности коммутаторов, процессоров и ассоциативной памяти согласованы, равны и все устройства работают с одним и тем же темпом П. В этом случае максимальная производительность системы будет равна ПNк. Фактически NC определяет количество операций, над которыми вычислительная система потока данных должна работать одновременно, чтобы была обеспечена ее 100%-ая загрузка.

Из этих соотношений видно, что чем меньше величина NC, тем более эффективно будет работать комплекс при малом параллелизме задачи.

Сравнивая эффективность загрузки традиционных машин с разрабатываемой, можно отметить два положительных фактора: какой бы большой ни была величина параллелизма n и какова бы ни была величина параллельно работающих каналов Nк, автоматически осуществляется предельно возможная загрузка системы. В традиционных системах с увеличением величин n и N возможности человека резко падают, начиная с N = 3-4. Этим можно объяснить то, что пользователи Connection Machine и nCUBE говорят лишь о 10% их средней загрузки на ориентированных на эти машины задачах.

Вторым преимуществом разрабатываемой системы является то обстоятельство, что в традиционных машинах параллелизм выявляется человеком или процессором на небольшом участке программы, не превосходящем, в лучшем случае, десятка команд. Поэтому средний параллелизм во времени исполнения для скалярных операций для таких машин как МВК "Эльбрус-2" не превосходит 2-3, а для векторных конвейерных машин типа Cray не превышает 10 [1,2]. Параллелизм в разрабатываемой машине автоматически определяется на данных всей задачи, находящейся в АП, в векторном ОЗУ и буфере, поэтому, чем больше память машины, тем большая задача может решаться на ней и тем большей будет величина параллелизма во времени, а, следовательно, и вероятность полной загрузки исполнительных устройств, и процент загрузки на участках задачи с малым параллелизмом. Немаловажным является и то обстоятельство, что в принятой нами концепции организации векторных вычислений параллелизм самих векторных вычислений на уровне операций над векторами определяется на скалярном процессоре по всей задаче, а параллелизм самой векторной операции над элементами вектора равен количеству элементов в векторе, длина которого может быть до 10 тысяч.

Концептуально, новая структурная схема машины, изображенная на Рис.6, больше отвечает характеристикам процессора, так как она не имеет операционной системы, автоматически на большом поле данных (объем АП) распараллеливает вычислительные процессы, обеспечивая при этом последовательность прохождения команд, синхронизируя их по данным. В дальнейшем, в отличие от обычных процессоров, будем называть его суперпроцессором.

Все это говорит о том, что можно ожидать достаточно высокой загрузки исполнительных устройств, а, следовательно, производительность одного суперпроцессора новой архитектуры будет соизмеримой с его максимальной производительностью, которую легко можно посчитать, зная предельный темп работы всей машины. Принимая темп работы равным 1 нс, что реально можно ожидать в недалеком будущем при использовании оптических принципов коммутации, при достаточно хорошей загрузке аппаратных средств можно получить максимальную производительность на скалярных операциях равную 10^{11} оп/с. Это на три порядка выше предельной производительности скалярного процессора, построенного на традиционных принципах. Предельная производительность, с учетом векторных операций и операций над массивами, может быть на один-два порядка выше.

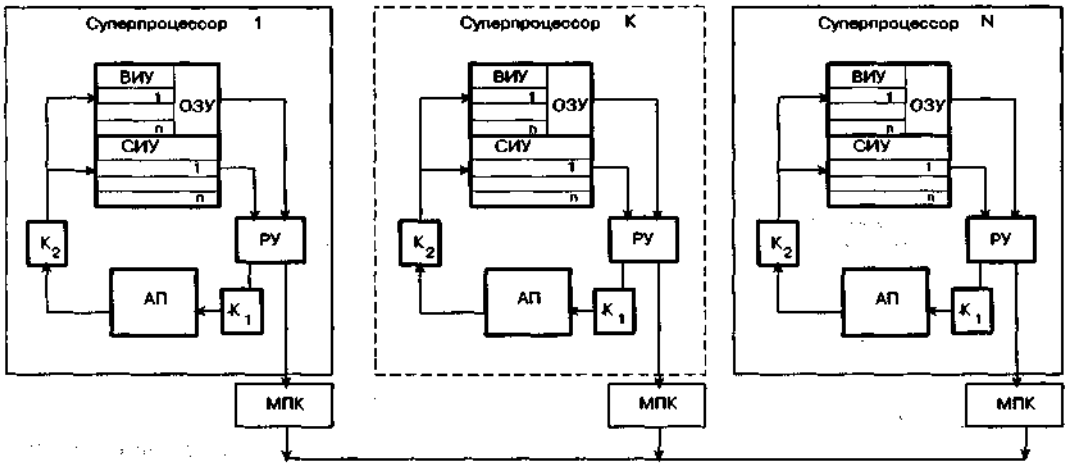


Рис.6а. Многопроцессорный комплекс на базе суперпроцессоров.

ВИУ- векторное исполнительное устройство, СИУ - скалярное исполнительное устройство, РУ - распределительное устройство, МПК - межпроцессорный коммутатор

Используя тот же принцип организации вычислительного процесса и рассматривая описанную выше систему потока данных, как один суперпроцессор, можно создать своеобразный многопроцессорный комплекс (Рис.6а). В этом случае определенные разряды адресации данных (I,П,T,NK) необходимо использовать для распределения токенов по суперпроцессорам. Очевидно, что такая многоступенчатая структура распределения токенов сначала между суперпроцессорами, а затем по модулям памяти внутри него, существенно замедлит вычислительный процесс только в том случае, если будут необходимы частые передачи данных, в особенности векторных, между суперпроцессорами. Если распределение задач между машинами возложить на человека, то можно ожидать, что в основном работа по распределению токенов между модулями памяти будет идти внутри машины. Безусловно, как и в традиционных комплексах, человеку справиться с этой задачей при числе суперпроцессоров более 10-ти будет достаточно сложно, несмотря на то, что ему придется распределять между ними достаточно локализованные крупные куски задачи. Подобные

многопроцессорные комплексы были исследованы на моделях и дали неплохие результаты. Это позволяет говорить о том, что максимальная произвдительность многопроцессорных вычислительных комплексов, построенных на принципах новой нетрадиционной архитектуры суперЭВМ, может достигать на полупроводниковой базе 10^{12} оп/с (темп = 10 нс), а с использованием оптических принципов коммутации и ассоциативной памяти - 10^{13} - 10^{14} оп/с и более (темп = 1 нс).

5. Общие соображения по выбору архитектуры машины

При принятых нами концепциях о выделении векторных операций в специальное исполнительное устройство со своей памятью и распределении командной памяти по всем исполнительным устройствам, укрупненная структурная схема представляется достаточно просто (Рис.6).

Производительность такого вычислительного кольца будет зависеть от темпа работы блока исполнительных устройств и их загрузки.

Темп блока исполнительных устройств можно считать неограниченным, так как можно практически неограниченно увеличивать количество скалярных исполнительных устройств. При достаточно большом числе каналов Nк, безусловно, скорость продвижения по кольцу могут сдерживать коммутаторы $К_1$ и $К_2$ но, учитывая принципиальную возможность их параллельной работы, а также дополнительные возможности оптических принципов коммутации, эту проблему можно решить. Наиболее принципиальным блоком, ограничивающим скорость продвижения данных по кольцу, является ассоциативная память (АП). Как показано в [3], в одном модуле невозможно получить требуемого быстродействия.

При требуемом объеме памяти, разбиение на модули ассоциативной памяти существенно повышает ее пропускную способность, но, в отличие от блоков исполнительных устройств и коммутаторов, имеет принципиальные ограничения:

- увеличение количества модулей при том же объеме памяти снижает объем каждого модуля и увеличивает вероятность его переполнения;
- как бы хорошо не была выбрана функция распределения данных по модулям, будут иметь место случаи одновременного обращения к одному модулю с нескольких направлений.

Поэтому, при реальном проектировании структуры машины потока данных за базу пропускной способности машины необходимо взять возможности реализации ассоциативной памяти, а все остальные блоки вычислительного кольца проектировать так, чтобы не сдерживать работы АП при условии минимизации величины NC по всему кольцу.

В настоящее время АП общим объемом в $10^{6\text{-}7}$ слов можно построить из 100 модулей, причем скорость работы каждого модуля будет не более 10 нс. Построить исполнительные устройства, работающие с темпом 10 нс, в настоящее время представляется возможным. Отсюда возникают требования к темпу работы коммутаторов - коммутация 100x100 каналов с темпом 10 нс. Реализовать такие требования по темпу работы коммутатора в одном модуле без использования оптических средств в настоящее время представляется

невозможным. Даже с использованием оптических средств коммутации, построить систему управления коммутаторами с требуемым быстродействием будет достаточно сложно и, возможно, придется использовать принцип "расслоения" работы коммутаторов.

При разбиении ассоциативной памяти на независимо работающие модули при выполнениии программ возникают ситуации "натыков", когда несколько исполнительных устройств обращаются одновременно к одному модулю АП. При этом возможна ситуация, когда вычислительный процесс не сможет идти дальше (так называемый "затыр"), если в кольце нет буфера объемом N слов. Этот буфер может стоять в любом месте кольца.

В потоковой модели вычислений нет переиспользования ячеек памяти, а с понятием общих данных связан процесс размножения данных. При выполнении процесса размножения в кольце рождается больше готовых пар, чем имеется станций обработки. Поэтому, для выполнения процесса размножения в общем случае также необходим буфер, причем емкость его должна быть большой (чтобы не накладывать ограничений на программирование). Этот буфер должен быть и быстродействующим. Поэтому, целесообразно иметь в кольцах небольшие быстродействующие буфера и общий буфер большого объема, связанный с буферами в кольцах.

Исходя из принципа минимизации NC, эти буфера целесообразно объединить в один. Так как токены, направляемые к одновходовым командам, не проходят через коммутатор ассоциативной памяти, а процесс обмена в системе буферов не должен быть привязан к номеру кольца, то такой буфер должен располагаться в кольце между выходом АП и входом ИУ.

Ранее указывалось, что в потоковой модели желательно соблюдать принцип "первый вошел - первый вышел". Если строго следовать этой идее, то в случае натыка по АП прием новых токенов на входные регистры коммутатора ассоциативной памяти должен быть заблокирован до окончания обработки всех поступивших ранее запросов. Так как мы не предусматриваем буфера между ИУ и АП, то при этом ИУ будут остановлены до окончания "разборки" натыка, а затем начнется последовательная обработка готовых пар, которые возникли в модуле АП при разборке натыка. Такая приостановка работы и последующий медленный разгон снижают производительность. Этого можно избежать, если на выходе АП поставить коммутатор-распределитель, который во время разборки натыка будет распределять возникающие готовые пары по всем ИУ. После завершения разборки натыка возникшие за это время готовые пары будут одновременно (параллельно) переданы на ИУ, чем достигается потеря всего лишь одного такта по сравнению с временем разборки натыка, когда не блокируется прием на входные регистры коммутатора АП.

При выполнении программ из-за натыков по модулям АП тормозятся, как правило, ИУ колец, в которых возникли токены, направленные коммутатором к одному модулю АП. При этом могут быть кольца, в которых ИУ в данный момент не заняты (в кольце готовых пар). Если отказаться от соблюдения принципа "первый вошел - первый вышел" в коммутаторе ассоциативной памяти, то для увеличения производительности необходимо в эти кольца передать готовые пары из колец, ИУ которых заняты, с помощью коммутатора- распределителя.

Коммутатор-распределитель должен учитывать занятость непосредственно ИУ, поэтому он должен стоять непосредственно перед ИУ.

Так как требование соблюдения принципа "первый вошел - первый вышел" не является жестким (т.е. может привести к успеху далеко не во всех возможных ситуациях), то рассматривается вариант архитектуры, в котором выходы модуля АП подключаются непосредственно ко входам коммутатора-распределителя, а его выходы к буферам готовых пар.

При разборке натыков блокируется прием токенов на все входы коммутатора АП

6. Описание работы одной из базовых схем

Рассмотрим несколько более подробно работу одного из вариантов базовой схемы (Рис.7).

Вычислительное кольцо состоит из следующих основных блоков: исполнительное устройство ($ИУ_{1\text{-}n}$), коммутаторов ассоциативной памяти (КМАП), модулей ассоциативной памяти (МАП), распределенного буфера (БУУ и БАУ), устройства регулятора коммутатора (УКМР) и коммутаторов распределителей (КМРУУ и КМРАУ).

Рассмотрим функции работы каждого блока кольца.

Исполнительное устройство.

Учитывая специфику выполнения команд управления, очевидно будет рациональным иметь, наряду со стандартным арифметическим микропроцессором (АМ), специализированный микропроцессор для выполнения управленческих операций (УУ). Каждое исполнительное устройство должно содержать память команд (ПК) с относительно большим временем записи и быстрым чтением команд. В каждую ПК должны быть записаны все команды выполняемой на машине задачи. На вход исполнительных устройств приходят пакеты готовых к выполнению операций (Рис.8а). Такой пакет состоит из "окраски" токена результата - разрядов индекса, разрядов итераций и разрядов активизации; номеров команды, куда необходимо направить результат, кода выполняемой операции (может отсутствовать), кода определяющего тип выполняемой команды и двух операндов.

В зависимости от типа выполняемой команды, данные поступают либо на входной регистр управления (вхУ), либо на входной регистр арифметической операции (вхА) (Рис.7).

С входных регистров данные передаются на УУ и АУ соответственно, если в пакете содержится код операции, то начинается выполнение операции, а одновременно с этим в ПК находятся коды операций следующих команд и адреса следующих команд.

Если придерживаться классической схемы потока данных, то вместе с кодом следующей команды в ПК хранится код операции, которая должна выполняться в настоящий момент. В этом случае, одновременно с кодом операции, должен храниться код типа команды, которая будет выполняться.

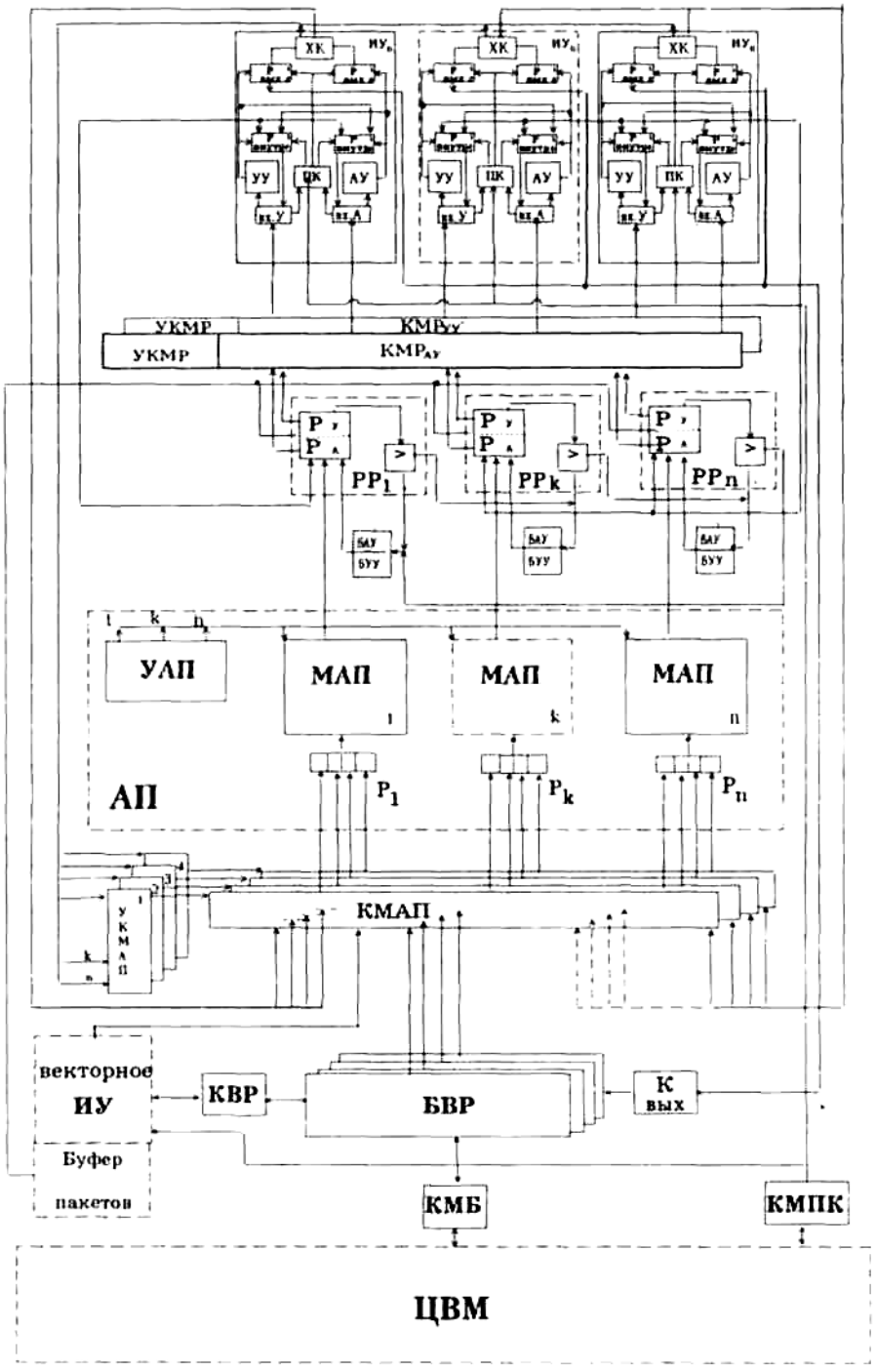


Рис.7. Принципиальная схема машины.

В случае классической последовательности, выполнение операции на ИУ должно начинаться со считывания команды из ПК для определения кода выполняемой операции (Рис.7).

По коду типа операции, которая должна выполняться следующей: двухвходовая или одновходовая, выдается результат либо на внешние два регистра $Р_{внеш1}$ $Р_{внеш2}$. либо на внутренние $Р_1$ и $Р_2$. Токен каждого выходного операнда формируется из данных результата, кода следующей команды, "окраски", состоящей из индекса, итерации, активации, кода хеширования, следующей операции и типа следующей операции (Рис.8б).

Код хеширования вырабатывается в устройстве хеширования (ХК), как правило, в результате простого или циклического сложения младших разрядов кодов I, T, П и NK. По семи разрядам хеш-функции определяется номер модуля ассоциативной памяти, где необходимо искать пару для сформированного токена. В том случае, если следующая команда должна быть одновходовой, выходной токен формируется на внутренних выходных регистрах $Р_1$ и $Р_2$. Токен, формируемый на этих регистрах, отличается от выходного токена только отсутствием кода хеш-функции. Токен с внутренних регистров передается, соответственно, либо на входные регистры вхУ, либо на регистр вхА.

Необходимо сформулировать условия, при которых ИУ готово к приему пакета из коммутатора КМР для выполнения операции.

Прием на любой из входных регистров возможен, если свободны выходные регистры (и внешние и внутренние) соответствующего микропроцессора и нет внутреннего запроса на выполнение операции. Таким образом, по каждому входу будут продолжаться вычисления до тех пор, пока не будет выполнена операция, требующая выхода на внешние регистры.

Такой принцип работы имеет определенные ограничения в части выполнения последовательности команд, имеющих один вход и два выхода. Как видно из структурной схемы ИУ, может быть выполнена последовательность не более чем из двух операций такого вида, так как третья операция не будет иметь места для размещения результата. Однако, это ограничение не столь существенно, если рассмотреть таблицу операций, приведенную на Рис.9.

Так, последовательность одновходовых команд 10-18, 28, 37, 38, 52 не имеет смысла. Остается команда 23, для которой целесообразно сделать специальное кольцо через внешние выходные регистры и коммутатор распределитель КМР, минуя ассоциативную память. В этом случае раздача параметра или константы во все указанные узлы графа будет выполняться почти одновременно всеми исполнительными устройствами.

Коммутатор ассоциативной памяти.

Токены (Рис.8в) из всех исполнительных модулей исполнительных устройств подаются на входы коммутатора, за исключением кодов хеш-функций. Последние подаются на соответствующие входы управления коммутатора АП (УКМАП). Функциональная схема основных коммутаторов (КМАП и КМР) идентична (Рис.10) и отличается только работой устройств управления. Каждый входной канал $К_{ВХ0Д1}$ при помощи элемента оптического транспаранта или аналогичным образом соединенных электронных вентилей, может быть соединен с выходным каналом $К_{ВЬ1Х2}$.

Код опер.	Тип команды	Номер след. команды	Индекс	Итерац.	Поколен. (активац.)	$Операнд_1$	$Операнд_2$
КОП	Тк	NKсл	I	Т	П	$Д_1$	$Д_2$

а)

Код $опер._1$	Код $опер._2$	Тип следующей команды $_1$	Тип следующей команды $_2$	Номер следующей команды $_1$	Номер следующей команды $_2$
$КОП_1$	$КОП_2$	Тк	Тк	NKсл	NKсл

Код следующей $опер._1$	Код следующей $опер._2$	Тип следующей команды $_1$	Тип следующей команды $_2$	Номер следующей команды $_1$	Номер следующей команды $_2$
$коп1_{сл}$	$КОП2_{сл}$	$Тк_1$	$Тк_2$	$Nк_1$	$Nк_2$

б)

Код операции	Тип след. команды	Номер след. команды	Хеш-функция	Индекс	Итерац.	Поколен. (активац.)	Операнд
КОП	Тк	NK	ХФ	I	Т	П	$Д_1$

в)

Код операции	Индекс	Итерац.	Поколен.	Номер команды	Колич. элементов
КОП	I	Т	П	NK	КЭ

команда - "фишка"

Номер БВР	Код опер.	Индекс	Итерац.	Поколен.	Номер команды	Колич. элементов
НБВР	КОП	I	Т	П	NK	КЭ

токен -"фишка"

Номер БВР	Код опер.	Индекс	Итерац.	Поколен.	Номер команды	Колич. элементов	Данное
НБВР	КОП	I	Т	П	NK	КЭ	$Д_2$

пакет-"фишка"

г)

Код опер.	Тип след. операции	Индекс	Итерац.	Поколен.	Номер след. команды	$Операнд_1$ Дескрипт. $вектор_1$	$Операнд_2$ Дескрипт. $вектор_2$
КОП	Тк	I	Т	П	NKсл	$Д_1$	$Д_2$

д)

Рис.8. Структура команд и данных.

Работа / Операции	УУ				АУ				Функции
	Вх дв		Вх од		Вх дв		Вх од		
	Выход		Выход		Выход		Выход		
	од	дв	од	дв	од	дв	од	дв	
1	+	-							
2	-	+							
3	+	+							
4	+	+							
5	+	-							
6	+	+							
7	+	+							
8	+	-							
9	+	-							
10	-	-	+	+					ВК(В1); А1А2:=(Д=К1)В1
11	-	-	+	+					ВП(В1); А1А2:=(Д=П1)В1
12	-	-	+	+					ВТ(В1); А1А2:=(Д=Т1)В1
13	-	-	+	+					ВИ(В1); А1А2:=(Д=И1)В1
14	-	-	+	+					ВС(В1); А1А2:=(Д=С1)В1
15	-	-	+	+					
16	-	-	+	+					то же, что оп 10-14
17	-	-	+	+					но "точно"
18	-	-	+	+					
19	+	+							
20	+	-							
21	+	-							
22	-	+							
23	-	-	-	+					ПДБ А1А2:=В1
24	+	-							
с 25 по 27					+	+			
28							+	+	ЛО А1А2:=(А1=Д)В1
с 29 по 36					+	+			
37							+	+	ОКР(В1) А1А2:=(А1=Д)В1
38							+	+	ОТБ(В1) А1А2:=(А1=Д)В1
с 39 по 46					+	+			
47	+	+							
48	+	+							
49	+	-							
50	+	-							
51	+	-							
52			+	+					НФ(В1) А1А2:={П=ДП(Д), Т1И1}В1
53	-	+							

Рис.9. Таблица операций.

Все элементы одной строки объединяются в выходной канал. Таким образом, эта матрица может одновременно коммутировать n входных каналов с n выходными. Управление элементами коммутации или вентилями в каждой схеме управления коммутатора осуществляется триггерами управления $T_У$ (Рис.11).

На схему управления УКМАП, как уже говорилось, подаются коды хеш-функций всех входных направлений. Этот код дешифруется дешифратором каждого направления, в результате чего каждое направление посылает запрос

на коммутацию с определенным направлением (возбуждается один из выходов дешифратора, который связан со схемой триггера управления соответствующего направления B_1, B_2).

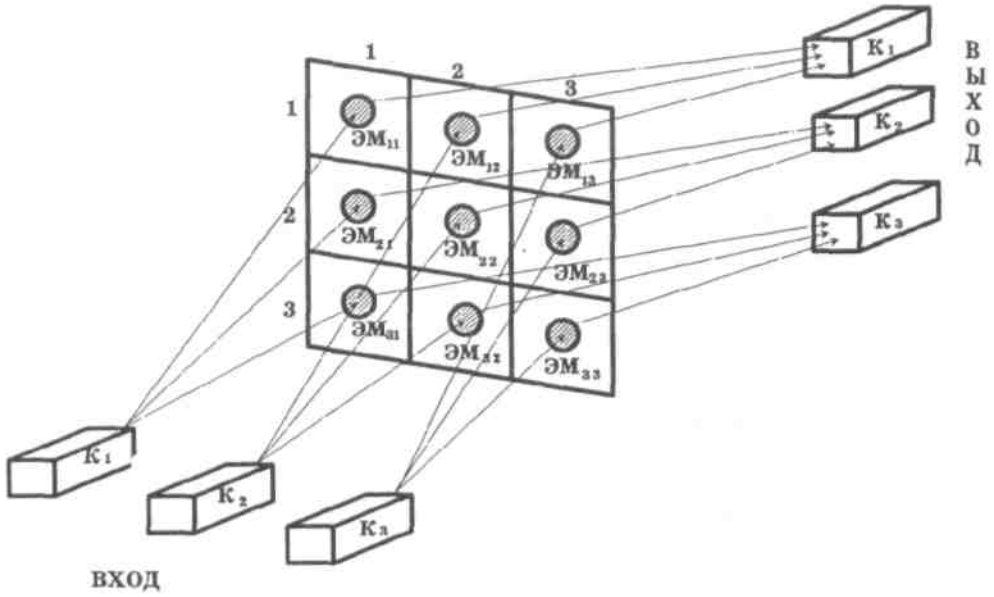


Рис.10. Типовая схема коммутатора.

Не исключена возможность, что несколько входных направлений запросят коммутацию с одним и тем же выходным направлением. Работа схемы состоит в том, чтобы все запросы к одному и тому же направлению обработать последовательно. Импульс пуска через соответствующие вентили B_1 опрашивает все входные запросы по каждому выходному направлению. По каждому выходному направлению происходит следующая работа. Если первое входное направление имеет запрос к каналу по выходу направления, то триггер $T_{укl}$ устанавливается в положение, открывающее коммутирующий элемент матрицы $ЭN_{кl}$ и устанавливает в нуль регистр хеш-функции этого направления.

Если запроса по этому направлению нет, то через инвертор И открыт вентиль $в_2$ и происходит опрос второго и т.д. направлений. Если по данному выходному направлению запросов нет или все обработаны, вырабатывается сигнал, устанавливающий триггер $T_{нк}$ в нулевое положение. После каждого пускового импульса происходит работа коммутатора, заключающаяся в передаче входных токенов по скоммутированным каналам на регистры $P_{1\text{-}n}$ модулей АП. После окончания передачи, начинается подготовка к новой работе элементов коммутационной матрицы. Если же не было "натыков" и за один такт работы коммутации все запросы были удовлетворены, следующий импульс пуска пройдет по всем вентилям $в_3$ и выйдет в виде конца цикла, который укажет на то, что следующая строка токенов по всем направлениям может быть принята

на входные регистры коммутатора (в частности, входными регистрами коммутатора могут служить выходные регистры ИУ).

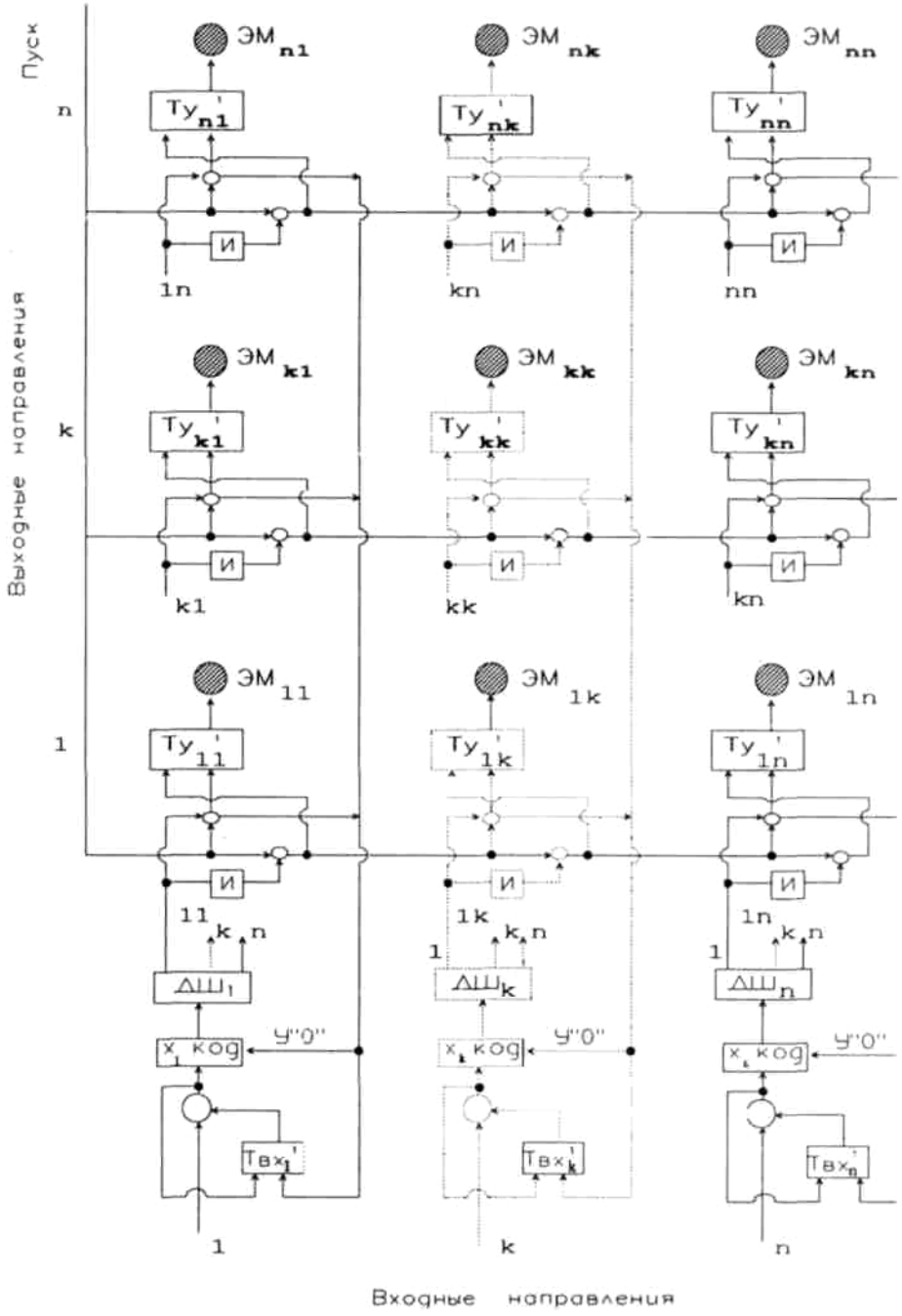


Рис.11. Коммутатор с последовательной обработкой.

Эта схема обладает тем недостатком, что пока вся строка не обработается, готовые токены должны ждать очереди, в то время как пути их коммутации могут быть свободными.

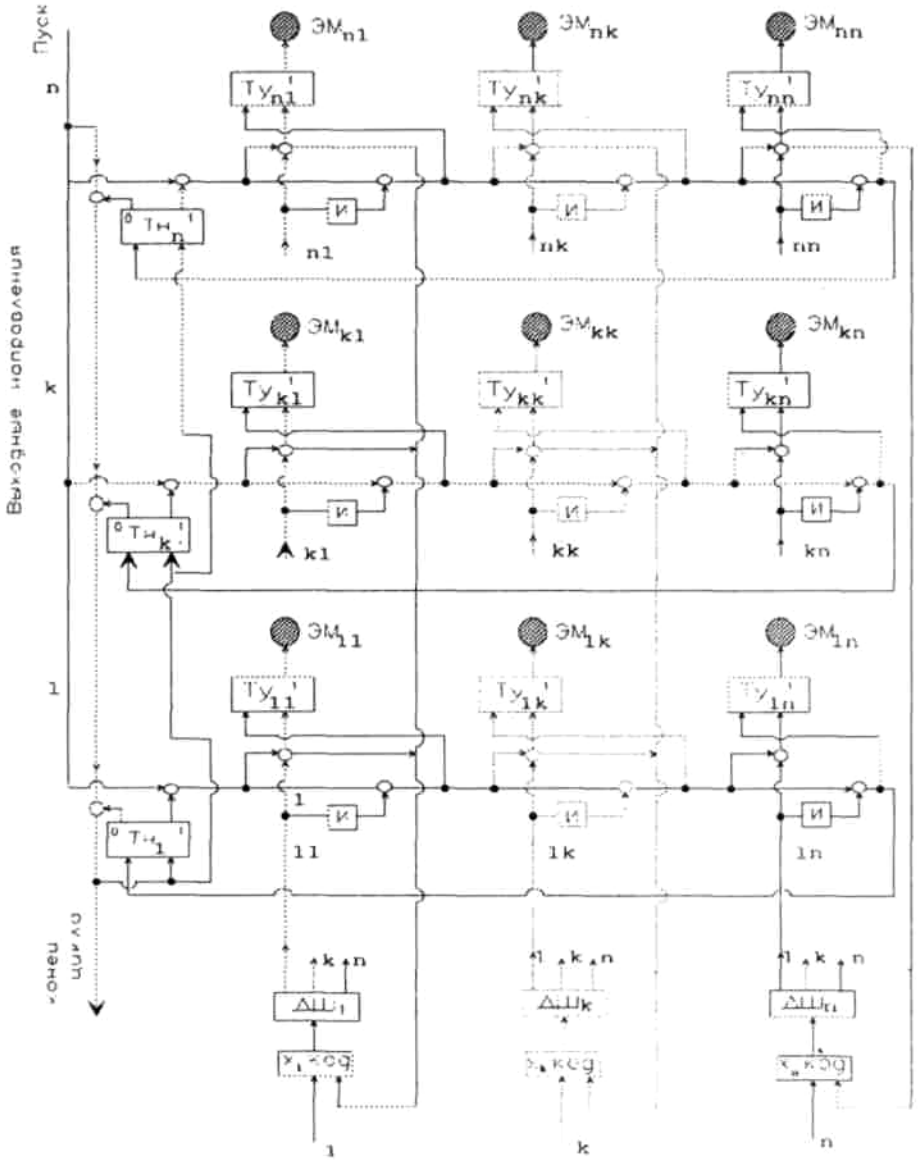


Рис.12. Коммутатор с асинхронной обработкой.

На Рис.12 приведена структурная схема УКМАП, позволяющая обрабатывать токены по мере освобождения канала по входу. Работа схемы по опросу

запросов входных каналов идентична предыдущей схеме, однако, после того, как запрос по какому-либо из входных направлений удовлетворен, одновременно с установкой в ноль кода хеш-функции устанавливается в единицу триггер занятости $Т_{вх}$ этого направления и на это направление может поступить следующий токен. И в той, и в другой схеме импульс пуска вырабатывается только тогда, когда выходные регистры коммутатора КМАП по всем направлениям свободны.

Отрицательной стороной этой схемы управления УКМАП (Рис.12) является то, что может не выполняться принцип работы первый пришел - первый пошел на выполнение. Действительно, не исключен случай, когда на одно и то же выходное направление есть запросы от нескольких входных направлений, а мы их обрабатываем в одной и той же последовательности. В результате чего не исключен случай, когда вновь пришедший на обработку токен будет обрабатываться раньше, уже стоящих на очереди направлений. Для исключения подобных ситуаций можно менять алгоритм работы схемы управления. Допустим, десять раз работать по алгоритму второй схемы и один раз по алгоритму первой.

Ассоциативная память.

Ассоциативная память выполняет несколько команд в зависимости от кода операции. Основные из них - поиск парного токена по ключу и считывание его в случае прохождения или запись вновь пришедшего токена на свободное место, если поиск закончился отрицательно. Как правило, считывание происходит со стиранием информации, но в некоторых случаях выполнение операций требуют сохранения считываемого токена. Считанная информация объединяется с вновь пришедшим токеном и образуют пакет (Рис.8д), который поступает на входные регистры коммутатора КМР (РУ и РА). Ключом, как правило, является код, состоящий из разрядов номера команды, индекса, итераций и активации. В случае ситуации, когда близок к переполнению один из АП, происходит перераспределение записи вновь пришедшего токена в модуль с наименьшим заполнением.

Управление равномерным заполнением модулей памяти осуществляет центральное управление ассоциативной памяти УАП. Запись в другой модуль требует изменения алгоритма считывания, замедляет общий темп работы АП, однако останов работы всей машины происходит только в том случае, если переполняется вся память АП. Подробный алгоритм работы АП описан в [4].

Работа распределителя регулятора и буфера памяти.

Пакеты объединенных токенов, готовые для выполнения операций ИУ, из АП передаются на входные регистры коммутатора распределителя через распределитель регулятор. На распределитель регулятор приходят пакеты токенов из трех источников:

- из ассоциативной памяти;
- из буфера, в котором пакеты хранятся отдельно для выполнения на арифметическом процессоре (БА) и для выполнения на процессоре управления (БУ);

- из устройства управления (пакеты для выполнения одновходовых операций дублирования).

Выходными устройствами РР являются:

- входные регистры КМР соответственно $КМР_{уу}$ - РУ и $КМР_{ау}$ - РА;
- регистры записи в буферную память либо свою, либо буфера следующего канала;
- входные регистры векторного исполнительного устройства.

Функции РР состоят в следующем:

- Распределение приходящих из ИУ и АП пакетов по каналам управления, арифметики и векторного исполнительного устройства на основании кода, определяющего тип выполняемой операции.
- Передача пакетов на входные регистры КМР в соответствии со следующим приоритетом ИУ, БУ, БА, АП.
- Запись пакетов в буфер, если входные регистры КМР заняты. Причем, запись в свой буфер производится в том случае, если он заполнен меньше, чем буфер следующего канала. В противном случае пакеты, не обработанные КМР, записываются в буфер следующего канала.
- В случае сигнала о возможности переполнения АП, операции, вызывающие рост параллелизма, откладываются в буфер, откуда бы они ни приходили (из буфера в том числе).

Буфер представляет собой регистровую память, в которой запись и считывание производятся по указателям счетчикам (Рис.13). Каждая запись по указателю записи увеличивает адрес счетчика на единицу. Так же работает и счетчик указателя считывания. Разность показаний счетчиков характеризует заполнение памяти и передается в соседний канал РР для принятия решения о том, в какой канал делать запись - свой или соседний.

Описанный принцип работы РР обеспечивает равномерное распределение готовых к выполнению пакетов по всем каналам и принятый алгоритм обработки пакетов первый пришел, первый обрабатывается. Фактически реализуется общая для всех каналов равномерно распределенная буферная память.

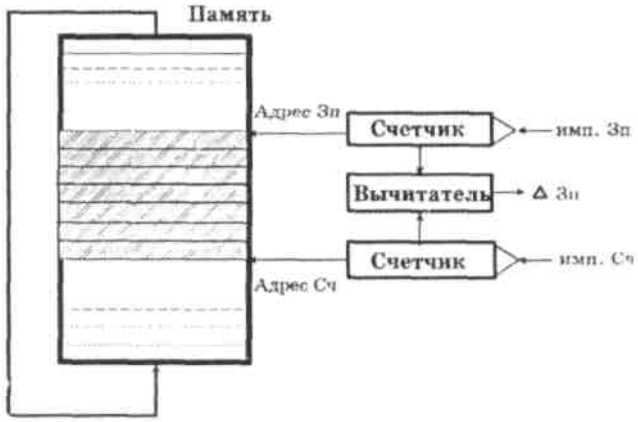


Рис.13. Буфер распределитель.

Коммутатор распределитель

Коммутатор распределитель построен на той же основе, что и КМАП (Рис.14). Отличие состоит в совершенно ином управлении триггерами T_y.

Коммутатор распределитель должен выполнять следующую функцию - распределить все пришедшие на входные регистры КМР пакеты по свободным входам исполнительных устройств. Коммутаторы КМРУУ и КМРАУ и их управление работают идентично и параллельно во времени, каждый по своим входным и выходным каналам.

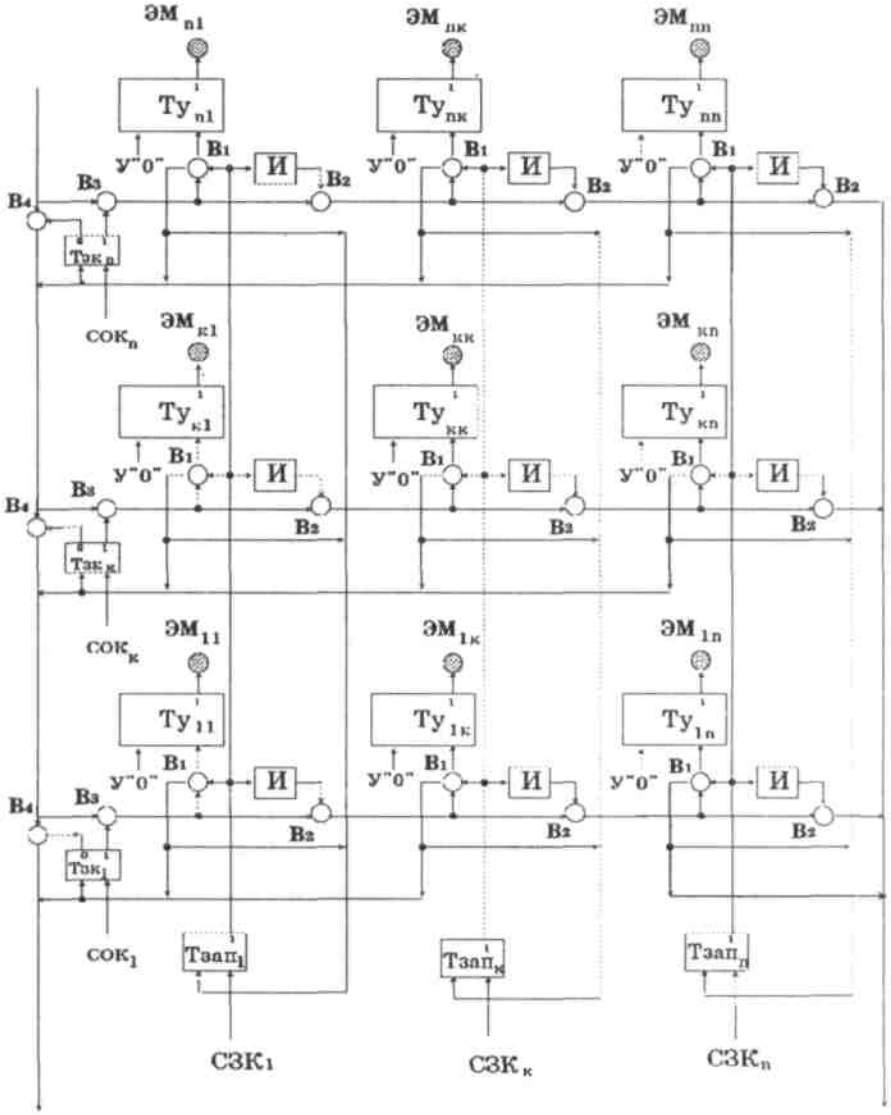


Рис.14. Коммутатор распределитель.

Основными управляющими элементами этой схемы являются триггера занятости входного канала по каждому направлению $T_{зк}$ и триггера запроса по входным направлениям $T_{зап}$ (Рис.14). Работа схемы начинается с импульса "пуск", который опрашивает занятость канала. Так, если триггер $T_{зкп}$ стоит в "0", импульс "пуск" проходит через вентиль B_4 и происходит опрос триггера занятости следующего канала $T_{зкк}$. Если входной канал не занят - $T_{зкк}$ стоит в "единице", опрашивающий импульс проходит через B_3 и вентили B_1, последовательно опрашиваются все направления по наличию запроса на входе. Если запрос есть на каком-либо направлении, срабатывает вентиль B_1 этого направления, триггер, управляющий элементами матрицы, встает в "единицу" ($T_{укк}$), тем самым соединяя свободное направление с соответствующим входным. Одновременно с этим вырабатывается импульс, устанавливающий в нулевое положение триггер запроса $T_{запк}$ и триггер занятости $T_{зкк}$ скоммутированных направлений. Этот же импульс приходит на опрос триггера занятости следующего выходного направления.

Возможны два окончания работы схемы управления:

- свободных направлений достаточно, чтобы удовлетворить все запросы;
- свободных каналов недостаточно для удовлетворения запросов по входам.

В первом случае, импульс пуска, проходя через вентиль B_3 свободного по выходу направления, проходит по всем вентилям B_2 этого направления и так как запросов на входе больше нет (все вентили B_2 открыты), вырабатывает сигнал конец цикла, говорящий о том, что все запросы удовлетворены. В том случае, если импульс пуска проходит через все вентили B_4, формируется сигнал конца цикла, говорящий о том, что ресурсы выходных сигналов исчерпаны, а запросы удовлетворены не полностью.

После работы схемы УКМР, работает система коммутатции и пакеты входных сигналов передаются на указанные триггерами Ту свободные направления.

Пакеты, для которых не нашлось свободных каналов, сбрасываются в буфер либо остаются на регистре в соответствии с алгоритмом работы устройства $РР_{l\text{-}n}$.

Существенным недостатком работы этой схемы является слишком большое время ее работы - максимальное время работы равно $n^2\tau_в$, где $\tau_в$ - задержка в вентиле измеряется сотыми долями наносекунды. Время работы этой схемы управления можно существенно уменьшить (более чем на порядок), если разбить матрицу управляющих триггеров на подматрицы и организовать двухкаскадную работу схемы по времени. В этом случае, каждой подматрице может соответствовать свой триггер управления подматрицы. Управление этими триггерами можно организовать по той же, ранее описанной, схеме. В этом случае триггера занятости запросов должны соответствовать занятости группы выходных триггеров и триггера запросов - состоянию группы триггеров запросов (Рис.14).

Работы этой схемы по времени разбивается на два последовательных пакета. Сначала коммутируются триггера, управляющие подгруппой, а затем идет аналогичная, ранее описанная работа управления в подгруппе. После проведения одного такого двойного цикла работы системы управления, могут быть проведены повторные циклы.

Сравним времена работы двухкаскадной и однокаскадной схем. Так, если необходимо коммутировать 128 на 128 направлений, матрицу можно разбить на 64 подматрицы (8x8 подматриц, каждая по 256 элементов). Время первого подцикла в этом случае будет (8x8) $\tau_в$, а время второго цикла будет (16x16) $\tau_в$. Таким образом, общий цикл двухступенчатого управления не превзойдет 310 $\tau_в$, в то время как максимальное время работы одноступенчатой схемы измеряется 16000 $\tau_в$.

Двухступенчатая схема имеет и то преимущество перед одноступенчатой, что лучше адаптируется к интегральной технологии.

Взаимодействие с векторным исполнительным устройством и центральной вычислительной машиной.

Векторное исполнительное устройство работает только с векторами (строками) размером в i28 элементов (обработка больших векторов происходит построчно). Поэтому, обмен информацией между векторным исполнительным устройством и скалярным вычислителем происходит как в обычном для исполнительного устройства виде (пакеты и токены), так и векторами. Пакеты в векторное исполнительное устройство (ВИУ) подаются из устройства $РР_1$ в буфер пакетов ВИУ, а токены из ВИУ поступают на входные регистры КМАП. Векторная информация, выходящая с ВИУ, поступает через коммутатор векторных регистров КВР в буфер векторных регистров. В КВР вектор преобразуется в вектор обычных токенов и поступает на входные регистры КМАП. Несколько сложнее обстоит дело с формированием вектора для передачи его из скалярного вычислителя в векторное исполнительное устройство. Дело в том, что различные элементы вектора формируются в АП в различные неопределенные времена. Поэтому, чтобы сформировать вектор, в систему запускается специальная команда "фишка". Эта команда "фишка" состоит: из кода операции, кода команды, индекса итерации, кода поколения, поля количества элементов. Из этой команды "фишки" формируется токен "фишка", состоящий из вышеперечисленных команды "фишки" и поля кода номера регистра БВР. Когда происходит объединение токен "фишки" с данным, образуется пакет "фишка", состоящий из полей токена "фишки" и поля данного (Рис.8г).

Поля I, T, П и NK служат для нахождения в АП элемента массива. Выполнение этой команды на исполнительном устройстве сводится к выдаче токена через коммутатор выхода $К_{вых}$ в БВР, прибавлении D к одному из полей I, T, П, проверке на окончание цикла и выдачи нового токена на КМАП. Во вновь сформированном токене поле одного данного пустое. После нахождения нового элемента вектора в АП, этот пакет с одним данным передается на свободное ИУ и цикл повторяется, пока все элементы вектора не будут сформированы. После того как сформирован вектор, из БВР он передается в векторное устройство, но на этом работа команды "фишки" не прекращается, т.к. массив, который мы хотим сформировать в ВИУ, может состоять из нескольких подвекторов по 128 элементов. Существование команды "фишки" заканчивается после того, как все подвектора данного вектора сформированы. По окончанию работы, команда "фишка" формирует пакет для ВИУ, который предписывает ему сформировать вектор в векторной оперативной памяти (ОЗУ).

Исполнением этой операции будет токен, который ВИУ передаст через КМАП скалярному вычислителю. Этот токен известит систему о том, что данный вектор готов к работе. Виртуальный адрес вектора будет указан в передаваемом токене в поле данных.

Осталось рассмотреть вопрос выполнения команды "фишки" в первом цикле ее работы, когда адреса БВР еще нет. Работа команды начинается с того, что она присваивает себе номер свободного регистра БВР. Если свободного регистра нет, необходимо эту команду отложить. В этом случае команда "фишка" отправляется в один из выделенных в БВР регистр буфера команд "фишек". Как только один из регистров БВР освобождается, первой на выходе команде "фишке" присваивается номер этого свободного регистра и она запускается в систему. Далее работа команды "фишки" идет по вышеописанному стандартному циклу.

Таким образом, взаимодействие всей скалярной вычислительной системы с ВИУ концептуально ничем не отличается от взаимодействия с исполнительным устройством с той только разницей, что в полях данных токенов и пакетов стоят не сами их данные, а описание векторов, их виртуальный адрес в ОЗУ ВИУ (Рис.8г).

Ввод и вывод данных в ЦВМ может проходить через те же регистры БВР путем пословного заполнения регистров как со стороны ЦВМ через КМБ с последующей выдачей информации на КМАП, так и со стороны скалярного вычислителя через $К_{вых}$ аналогично тому, как это выполняется при формировании векторов. Ввод программы производится через коммутатор памяти команд КМПК на все ПК исполнительных устройств, одновременно включая ПК векторного исполнительного устройства.

7. Возможные варианты упрощения и развития структурной схемы машины потока данных

Прежде всего необходимо обратить внимание на то, что приведенные структурные схемы будут успешно работать, если в систему команд ввести команды, позволяющие работать с результатом предыдущей операции.

В этом случае, все арифметические команды могут быть как одновходовыми, так и двухвходовыми, что существенно может разгрузить АП и повысить эффективность выполнения, так называемых, сильно связанных участков программ.

Дальнейшее развитие оптимизации сильно связанных команд может привести к введению регистров внутри процессора. Возможность использования в предшествующей команде результата предыдущей команды, позволяет легко решить проблему ввода константы из памяти ПК в поле данных без расширения формата команды и т.д.

Однако, все изменения в системе команд хотя и не требуют существенных изменений в структуре процессора и машины в целом, должны быть детально изучены и промоделированы. Поэтому, для первого макетного образца возможно целесообразно будет в качестве исполнительных устройств использовать транспьютеры. В транспьютере есть собственная командная память, все

специальные команды процессора потока данных могут быть достаточно эффективно интерпретированы в командах транспьютера. Исследование ввода регистров и специальных экстракодов для вычислений функций по одному-двум параметрам и другие возможности эффективной обработки сильно связанных участков программ может с успехом проводиться на системе с использованием транспьютера для имитации работы исполнительных устройств (Рис.15). Безусловно, транспьютеры будут работать с меньшим быстродействием, чем специально разработанные для этих целей микропроцессорные наборы. Этот недостаток может быть в какой-то мере скомпенсирован увеличением числа транспьютеров внутри каждого исполнительного устройства.

8. Оценка производительности скалярного процессора на электронной элементной базе

Как уже говорилось, основным звеном, сдерживающим производительность машины потока данных, является ассоциативная память. Если реализовать модуль памяти на интегральных схемах БИКМОП структуры с разрешающей способностью технологического процесса 0,7 микрон, то можно говорить о темпе работы, период которого не превышает 10 нс [4].

Будем считать, что модуль такого порядка можно реализовать объемом в 32 тыс. ключей. Из ста модулей такой памяти можно иметь достаточный объем АП (более трех миллионов слов). Коммутатор-распределитель можно набирать из однотипных модулей для того, чтобы обеспечить необходимый средний темп для данных, поступающих из АП. Очевидно, двумя-четырьмя коммутаторами распределителями можно обеспечить необходимый темп обработки поступающих данных. При этом в каждом коммутаторе может одновременно обрабатываться два пакета по каждому направлению.

С таким темпом может справиться работа 2-4 транспьютеров, в каждом из которых может обрабатываться по два пакета одновременно в конвейерном режиме. Четыре коммутатора КМАП в построчном режиме обеспечат необходимый темп работы для АП.

Таким образом, максимальная производительность $П_{NK} = 10^{10}$ операций в секунду будет реализована только в том случае, если параллелизм задачи на объеме памяти в 3 млн. слов будет не меньше

$$NC = N_{КаН} (N_{АП}C_{АП} + N_{КМР}C_{КМР} + N_{ИУ}C_{ИУ} + N_{КМАП}C_{КМАП})$$

Напомним, что NC - это общее количество операций, которое должно выполняться в системе при ее полной загрузке; $N_{КаН}$ - число каналов системы; $N_{АП}$, $N_{ИУ}$, $N_{КМР}$, $N_{КМАП}$ - число однотипных устройств в блоке каждого канала; $C_{АП}$, $C_{КМР}$, $C_{ИУ}$, $C_{КМАП}$ - уровень конвейеризации каждого устройства.

Так, для обеспечения темпа продвижения информации по кольцу, на современной технологической базе необходимо иметь $N_{АП}=1$, $N_{КМР}=2$, $N_{ИУ}=2$, $N_{КМАП}=4$ и соответственно $C_{АП}=2$, $C_{КМР}=2$, $C_{ИУ}=2$, $C_{КМАП}=1$, что соответствует времени операции АП=20 нс, времени операции КМР=40 нс, времени операции на транспьютере 40 нс и времени операции КМАП=40 нс. Общая максимальная производительность системы каналов при полной загрузке будет 10^{10} оп/с. Полная загрузка системы будет на участках задачи, где параллелизм

вычислительного процесса превосходит величину n = NC = 1300. Во всех остальных случаях загрузка будет пропорциональна $К_З$ = n / NC.

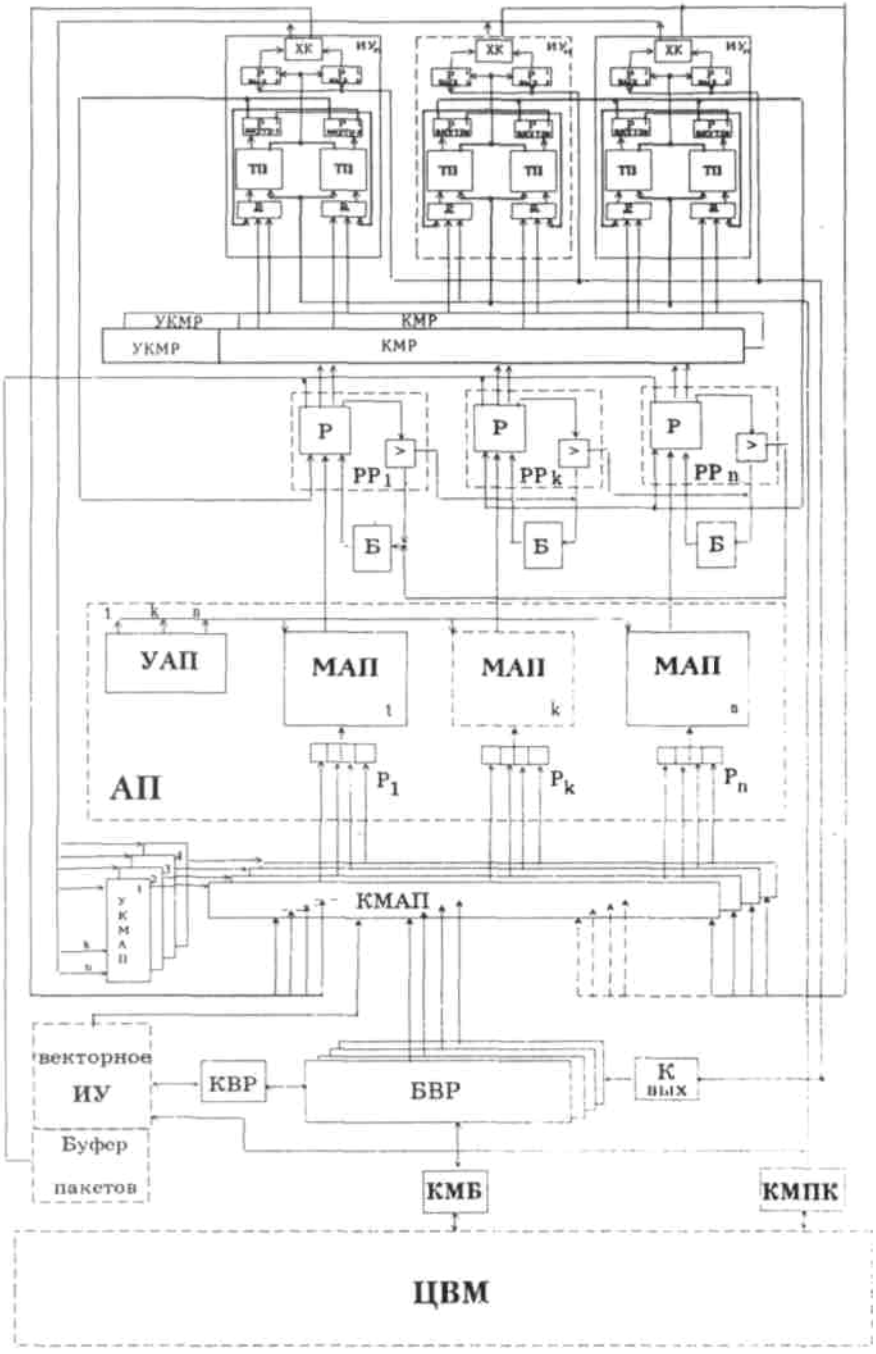


Рис.15. Модернизированная структурная схема машины.

Чтобы понять, насколько критично требование параллелизма задачи n = 1300, проведем аналогичный анализ для известных нам традиционных систем. Ясно, что для машин типа CM-5 и NCUB, эта величина будет на два порядка выше, а для конвейерных машин типа CRAY того же порядка. Однако, необходимо учитывать, что для машин, работающих по принципу потока данных, требование по полной загрузке является необходимым и достаточным, в то время как для традиционных машин вопрос загрузки ресурсов машины и корректности выполнения задачи во времени всецело зависит от человека.

Оценка производительности и загрузки машины потока данных проводилась без учета возможностей векторного исполнительного устройства, загрузка которого осуществляется от скалярного процессора, что может увеличить производительность комплекса более чем на порядок.

Выводы

1. Состояние развития суперЭВМ на настоящем этапе характеризуется тем, что производительность однопроцессорных систем, работающих на традиционных принципах, близка к своему физическому пределу, что на два-три порядка ниже требуемой как на скалярных, так и на векторных операциях.

2. Решение проблемы достижения требуемой производительности для решения больших сложных задач лежит на пути создания высокопараллельных систем обработки данных. Оно входит в противоречие с принципами работы и организации вычислительных процессов традиционных архитектур суперЭВМ. В традиционных высокопараллельных структурах задача распределения параллельно работающих вычислительных средств по параллельным процессам внутри задачи или между задачами возложена на человека. При увеличении числа параллельных вычислительных структур человек не в состоянии справиться с этой задачей.

3. Принцип организации вычислительного процесса, заложенный в структурах управления данными, решает эту задачу.

4. Реализация архитектуры суперЭВМ, работающей по принципу управления данными, наталкивается на целый ряд серьезных проблем, к которым, в первую очередь, относятся: организация работы специальной оперативной памяти объединения данных, повторная входимость программ, рекурсии, итерации, циклы, работа со структурами данных, инвариантность кода команд, работа с константами, чистка "мусора", информационные "взрывы" и т.д.

5. Зарубежные специалисты работают в этом направлении достаточно широким фронтом. Имеется порядка десяти проектов, в которых решаются все вышеперечисленные проблемы, однако высокой эффективности работы супер-ЭВМ на этом принципе в настоящее время достичь не удалось. Этим, очевидно, сдерживается их внедрение.

6. В рассмотренной новой архитектуре суперЭВМ основные проблемы построения машин, работающих на принципе потока данных, решены на достаточно высоком программном и схемотехническом уровнях, которые позволяют на современной технологической базе реализовать производительность однопроцессорной суперЭВМ на один - два порядка выше существующих. Даны

предложения по реализации многопроцессорного комплекса на тех же принципах.

7. Одной из отличительных особенностей структуры суперЭВМ является организация оперативной памяти объединения данных, которая является определяющей в достижении предельной производительности.

Предложено разделение памяти на командную, векторную и скалярную (ассоциативную).

Решен вопрос модульной организации памяти объединения данных.

Впервые предложен метод организации параллельного выполнения векторных операций с помощью скалярного процессора.

Достигнута высокая пропускная способность единой инвариантной памяти команд и констант.

8. Новая архитектура достаточно хорошо адаптирована к возможностям оптики, в особенности в устройствах ассоциативной памяти и коммутаторах, которые составляют основу суперЭВМ.

9. Наряду с решением наиболее важного вопроса - автоматического распределения ресурсов вычислительных средств с дискретностью до операции, новая архитектура суперЭВМ обладает целым рядом весьма полезных принципиальных свойств:

- ассоциативная память фактически берет на себя функции организации вычислительного процесса, ввиду чего исключается самая нерегулярная часть структуры суперЭВМ. Это обстоятельство делает архитектуру машины чрезвычайно регулярной, обладающей высоким коэффициентом применяемости элементов, что позволяет использовать для ее изготовления технологические средства, реализующие предельную интеграцию элементно-конструкторской базы;
- обладает свойствами толерантности, что чрезвычайно важно как для этапа изготовления, так и для этапа эксплуатации;
- исключает необходимость функций операционной системы для распределения как памяти, так и процессоров.

10. Разработанная архитектура с предложенными новыми технологическими решениями основных устройств позволяет существенно расширить пределы производительности суперЭВМ за счет аппаратного распараллеливания вычислительных средств при решении больших задач, обладающих внутренним параллелизмом.

Литература

1. Бурцев В.С. Анализ результатов испытаний МВК "Эльбрус-2" и дальнейшие пути его развития. М., 1988. (Препринт ОВМ АН СССР N 208)

2. Бурцев В.С., Кривошеев Е.А., Асриэли В.Д., Борисов П.В., Трегубов К.Я. Векторный процессор МВК "Эльбрус-2" СуперЭВМ. М.: ОВМ АН СССР,1989. С. 3-18.

3. V.S.Burtsev, V.B.Fyodorov. Associative Memory of New Generation Supercomputers Based on Optical Information Processing Principles.

Holography and Optical Information Processing, 1991, Vol. 1731, P. 201-206.
4. Проект ОСВМ. 1993.
5. Бурцев В.С. Оптические принципы обработки информации в архитектуре суперЭВМ. М.,1992. (Препринт ВЦКП РАН, N 24, С. 27-36).
6. Бурцев В.С. Тенденции развития суперЭВМ. Вычислительные машины с нетрадиционной архитектурой суперЭВМ. М.: Наука, 1990. Сб.1. С.3-26.
7. Popadopoulos G., Culler D. "Monsoon: an Explicit Token-Store Architecture", Sigarch Computer Architecture News , 1990, Vol.18, N.2.
8. Culler D. "The Explicit Token Store", Journal of Parallel and Distributed Computing, 1990, Vol.10, P. 289-308.
9. Popadopoulos G., Traub K. "Multithreading: A Revisionist View of Dataflow Architectures", Sigarch Computer Arch. News, 1991, Vol.19, N.3.

Использование оптических средств обработки информации в архитектуре суперЭВМ

В.С.Бурцев

Аннотация Предпринята попытка определить возможность использования оптических принципов обработки информации в вычислительных средствах. Произведен краткий сравнительный анализ возможности использования оптических методов обработки информации в различных стандартных узлах суперЭВМ нового типа. Показана возможность построения суперЭВМ с использованием оптических средств в 30 - 60% оборудования суперЭВМ.

Введение

Использование оптических средств обработки информации необходимо оценивать в сравнении с возможностями реализации тех же функций различных устройств на полупроводниковой элементной базе. Рассмотрение целесообразности использования тех или иных средств необходимо проводить с учетом не только сегодняшнего состояния развития архитектуры суперЭВМ, оптики и полупроводниковой техники, но и анализом их развития в будущем.

Имея в виду преимущества оптических средств перед полупроводниковыми электронными в части их широкополосности и широкоформатности при передаче информации, необходимо найти такие области их применения, где они были бы совершенно необходимыми. По нашему мнению, они, в первую очередь, могли бы найти применение в новом поколении суперкомпьютеров.

Однако, было бы неправильным не отметить и достаточно принципиальные недостатки оптических средств на настоящий момент времени. Основные из них следующие:

-оптические логические элементы (матрицы вентилей) имеют значительно большее энерговыделение по сравнению с полупроводниковыми;

- отсутствует запоминающий элемент, обладающий малым временем доступа Топ, пригодный для создания регистров кратковременного хранения и оперативной памяти больших размеров;
- затруднена интеграция устройств, например, устройств управления, обладающих большим количеством одиночных связей ввиду ограничений, связанных с допустимым радиусом изгиба световода.

Создание суперЭВМ на чисто оптических принципах обработки информации позволит более полным образом использовать достоинства оптических средств в деле создания дискретной высокопроизводительной вычислительной техники. Любые гибридные электронно-оптические способы построения устройств и комплексов в целом приводят к неоправданным потерям в быстродействии и мощности при переходе от оптики к электронике и обратно. Кроме того, несмотря на высокие достижения электроники в части быстродействия (100 пикосекунд задержки на вентиль), ограничения электронных средств по быстродействию не позволяют в полной мере использовать такие особенности оптических принципов обработки информации, как высокая широкоформатность и широкополосность передачи и коммутации сигнала.

К сожалению, состояние научно-технических исследований на настоящий момент не позволяет ставить вопрос о переходе к опытно-конструкторским работам по созданию универсальной сверхбыстродействующей вычислительной системы дискретного действия, работающей полностью на оптических принципах.

1. Перспективы развития архитектуры суперЭВМ

Современной тенденцией в разработке суперЭВМ является повышение их производительности до 10^{12} операций в секунду (оп/с) при емкости оперативной памяти порядка 10^9 слов ($\sim 10^{11}$ бит). Как указывается в [1] пределом производительности процессора, реализованного на полупроводниках и с использованием при организации вычислительного процесса принципа фон-Неймана, является производительность, равная 10^8 оп/с на скалярных вычислениях и 10^9 оп/с на векторных вычислениях. Единственным способом повышения производительности вычислительной системы является увеличение количества процессоров, то есть использование многопроцессорных и многомашинных вычислительных систем. Однако, их увеличение, несмотря на то, что они работают в единой системе, вызывает непреодолимые трудности при программировании задач для таких систем даже в тех случаях, когда вычислительный процесс может быть легко распараллелен на математическом уровне [2].

Критическая ситуация возникает когда параллельный алгоритм, описанный на уровне численных методов, программист должен воплотить в программу. Дело в том, что программа предполагает для выполнения параллельных участков программ использование определенных ресурсов вычислительных средств (процессора, памяти, каналов, дисков и т.д.). Поскольку задействованы ресурсы, возникает вопрос в течение какого времени они используются. Программисту необходимо точно знать время выполнения отдельных параллельных процессов, которые часто зависят от данных, и распределить вычислительные

ресурсы таким образом, чтобы свести к минимуму потери времени. Очевидно, что при решении указанных выше сложных задач человек не способен правильно программировать существующие параллельные процессы параллельно и во времени. Такая функция может быть выполнена только оборудованием, имеющим информацию о динамике вычислительного процесса в каждый момент времени. Именно благодаря этому, необходима новая архитектура суперЭВМ, отличная от архитектуры фон-Неймана.

Основными особенностями организации вычислительного процесса в новой архитектуре суперЭВМ являются:

- на уровне программирования нет необходимости определения временной последовательности выполнения операции, она начинается сразу же, как только собраны все данные, требуемые для ее выполнения;
- концептуально нет памяти (нет оператора присваивания);
- данные от оператора к оператору передаются по связям, организующим вычислительную сеть.

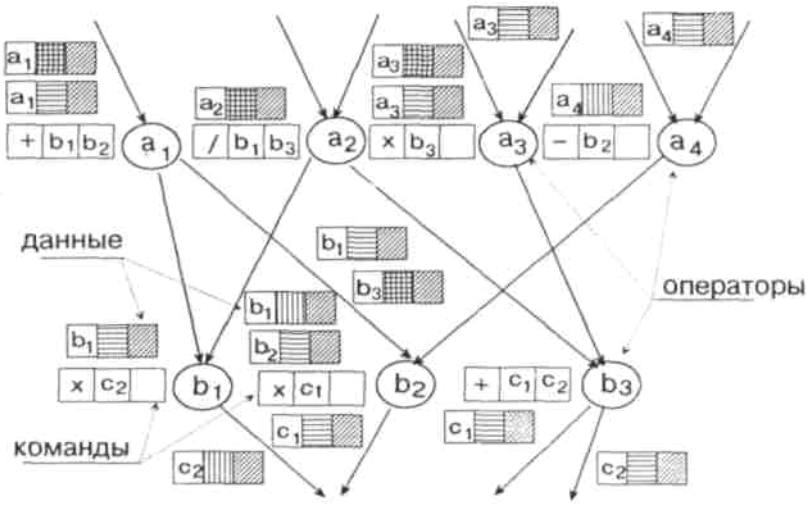


Рис.1. Граф вычислительного процесса.

Любой вычислительный процесс, как показано на Рис.1, может быть представлен в виде графа, в вершинах которого стоят операторы, производящие действия над данными, а данные по мере их обработки операторами по дугам перемещаются к следующему оператору. Таким образом, в отличие от принципа фон-Неймана, где операторы и данные вызывались последовательно, в новой архитектуре осуществляется параллелльная обработка данных по мере их поступления в операторы.

Команда новой машины (Рис.2) состоит из указания действия над данными и указателя на следующий оператор (узел), к которому необходимо передать результат обработки. Данные же, в свою очередь, имеют указатели, к какому узлу поступить на дальнейшую обработку, и окраску, которая позволяет идентифицировать данные одного комплекта, поскольку одной и той же программой могут пользоваться несколько комплектов данных.

Команда

Код операции	Адрес следующей операции	Адрес следующей операции
+	b_1	b_2

Данные

Адрес следующей операции	код окраски	Код операции	Данные
a_1		+	D

Пакет на выходе из АП

Адрес следующей операции	Код окраски	Код операции	Данные	Данные
C_2		+	D_3	D_5

Рис.2. Структура команд и данных.

Несложно представить себе блок-схему новой суперЭВМ (Рис.3), отвечающей описанному выше принципу обработки данных.

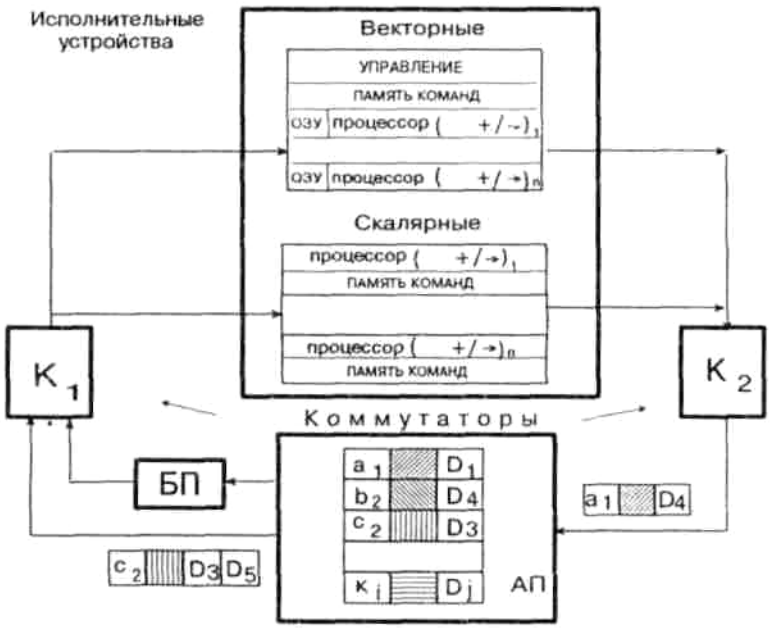


Рис.3. Блок-схема супер-ЭВМ нового типа.
АП - ассоциативная память, К - коммутатор.

Основной цикл работы такой машины сводится к следующему:
- после того как данные найдут себе пару и необходимую команду в ассоциативной памяти (АП), они поступают на коммутатор $К_1$, который передает их на любое свободное исполнительное устройство. Исполнительное устройство выполняет операцию в соответствии с указанием команды, и результат

вместе с адресом следующей команды отправляет на коммутатор $К_2$, который передает их в ассоциативную память.

В том случае, если в АП находятся данные с искомым ключом, то они считываются, и объединенный комплект данных (как правило два) выдается для выполнения следующего оператора на одном из исполнительных устройств. Считывание данного (или данных) из памяти в этом случае сопровождается его (или их) стиранием. Если же парного данного по ключу не найдено, то данное вместе с ключом записывается в свободную ячейку АП.

Для того, чтобы максимальная производительность такой вычислительной системы была не ниже 10^{11} оп/с, необходимо реализовать передачу данных по "кольцу", содержащему 100 скалярных исполнительных устройств, с частотой не менее 1 ГГерца. Произведенью оценки возможности реализации АП емкостью 10^9 слов и работающей с частотой 1ГГц говорят о том, что АП является самым узким местом в системе. Поэтому, в приведенной структуре суперЭВМ приняты следующие меры, позволяющие решить эту проблему:

- память должна быть модульной (емкость модуля обратно пропорциональна скорости его работы);
- введены специальные векторные исполнительные устройства, работающие со своей векторной оперативной памятью (Рис.4). Векторная оперативная память работает по принципу обычной прямоадресуемой памяти и должна обладать высоким темпом работы и может иметь относительно большое время доступа. В ассоциативной памяти хранятся только скаляры и описатели векторов;
- на случай информационного "взрыва" системы данные, готовые к выполнению, хранятся в буферной памяти (БП);
- введена система регулирования загрузки памяти - команды, приводящие к существенному заполнению памяти, могут задерживаться по выполнению;
- векторное исполнительное устройство содержит специализированный процессор, позволяющий избежать пересылок больших массивов данных между векторными исполнительными устройствами ассоциативной памяти;
- командная память выделена в отдельные блоки, не входящие в ассоциативную память.

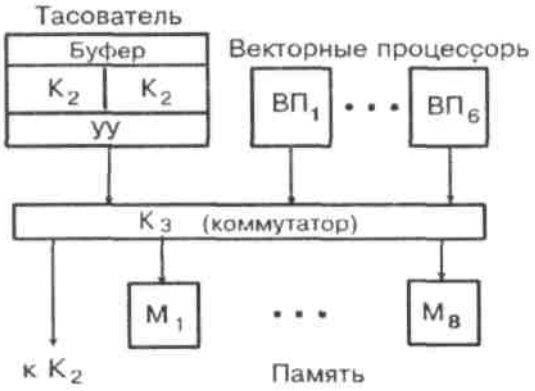


Рис.4. Векторное исполнительное устройство.

Векторное исполнительное устройство оперирует страницами, содержащими 128 слов. Каждый из 8 модулей оперативной памяти состоит из 8192

страниц. Максимальный размер вектора 128 страниц. Имеется восемь векторных процессоров - страниц: семь арифметических и один специализированный - "тасователь". Каждый арифметический процессор содержит 128 процессоров, оперирующих с 64 битовым кодом. Тасователь осуществляет перестановки элементов вектора по заданным индексам другого вектора, маскирование векторов, а также наложение векторов с преобладанием элементов одного из них. Коммутатор 8x8 направлений за несколько десятков наносекунд коммутирует устройства, после чего с темпом 1ГГц осуществляет обмен данными каждого устройства с каждым.

Наряду с этим новый архитектурный принцип предъявляет более высокие требования к широкополосности и широкоформатности основных устройств: скалярных и векторных исполнительных устройств, ассоциативной памяти и коммутаторов. Каждое из них обычно характеризуется двумя временными параметрами: временем выполнения операции $T_{оп}$ и темпом выполнения операции $T_{темп}$. Первый из них, в основном, определяется временем задержки в прохождении информации внутри устройства; а второй - темпом работы логических элементов устройств.

Новый принцип организации вычислительного процесса в некоторой степени снижает требования к параметру $T_{оп}$, перемещая центр тяжести на скорость выполнения операции $T_{темп}$.

В этом плане структура является шагом в направлении использования оптических принципов обработки информации, поскольку для оптики значительно сложнее конкурировать с электроникой в снижении времени выполнения операции Топ, чем в увеличении темпа выполнения операций, где основные преимущества оптики, широкая полоса и широкий формат, играют основную роль.

Переход к описанной выше структуре суперЭВМ обеспечивает решение ряда важных проблем создания сверхвысокопроизводительных вычислительных устройств:

- благодаря аппаратной реализации одновременного прохождения на вычислительных средствах параллельных процессов, заложенных в алгоритме задачи, существенно расширяется предел производительности суперЭВМ;
- человек освобождается от распределения ресурсов вычислительных средств при программировании;
- функции управления в организации вычислительного процесса реализуются с помощью ассоциативной памяти, имеющей регулярную структуру, вместо использования для этой цели устройств управления различной структуры;
- аппаратно решается функция операционной системы по распределению оперативной памяти;
- новая структура устойчива к отказам и сбоям, что дает возможность, например для ассоциативной памяти, оставаться работоспособной при достаточно большом проценте неисправных ячеек;
- приведенная структура аппаратно реализует параллелизм вычислительного процесса с дискретностью до операции, как над скалярами, так и над векторами и матрицами. Та же система может работать, реализуя параллелизм на уровне процедур. В этом случае, в качестве исполнительных устройств могут быть использованы транспьютеры, выполняющие более сложные процедуры, чем одна операция;

- при расширении класса задач и переходе к обработке в качестве данных сложных структур (деревья, списки, графики, сети и т.д.) требования к реализации массового параллелизма вычислительного процесса становятся более жесткими [3].

Исследования, проведенные в Вычислительном Центре Коллективного Пользования (ВЦКП) в области параллельных вычислений на ассоциативных сетях, представляют собой следующий шаг в направлении реализации массового параллелизма в архитектуре вычислительных средств.

Ниже будет произведен сравнительный анализ возможности создания устройств суперЭВМ новой архитектуры.

2. Реализация оптических принципов обработки информации в суперЭВМ с нетрадиционной архитектурой

2.1. Скалярное исполнительное устройство

В настоящее время в одном корпусе на полупроводниках может быть реализовано многофункциональное исполнительное устройство с темпом работы 5-10 нс и временем выполнения операции порядка 50 нс.

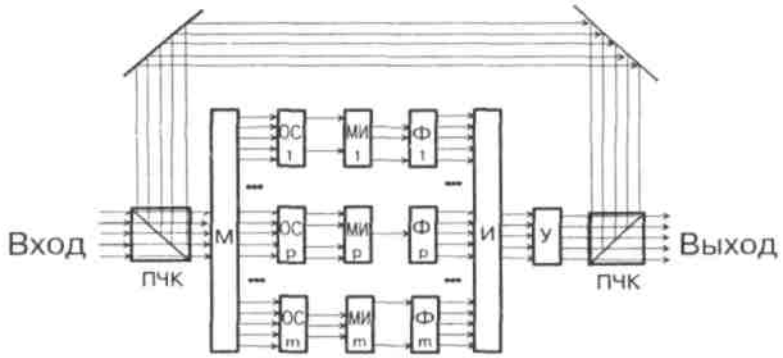


Рис.5. Типовая схема оптического процессора.

ПЧК - поляризационно-чувствительный кубик, М - мультиплексор, И - система суперпозиционирования изображения, ОС - блоки определения символов, МИ - матрицы световых инверторов, А - матрица усилителей, Ф - формирователи измененных символов.

Оптический процессор, типовая схема которого показана на Рис.5, неконкурентоспособен с полупроводниковым по основному параметру - работе, затраченной на одно логическое срабатывание $A_э = \tau_з P_э$, где $\tau_з$ - задержка логического элемента, $P_э$ - потребляемая им мощность.

Реализованное в 1989 году в Bell Lab аналогичное арифметическое устройство на CC Sid не превосходит по темпу 1 мкс и потребляет значительно большую мощность на одно логическое срабатывание.

2.2.Память

Более 15 лет ведутся поиски материала, пригодного для реверсивной оптической памяти. Существующие материалы оптической памяти по времени записи

в сотни раз проигрывают полупроводниковым. В настоящее время представляется возможным реализовать на оптических принципах динамическую память, как в свободном пространстве, так и на оптоволокне.

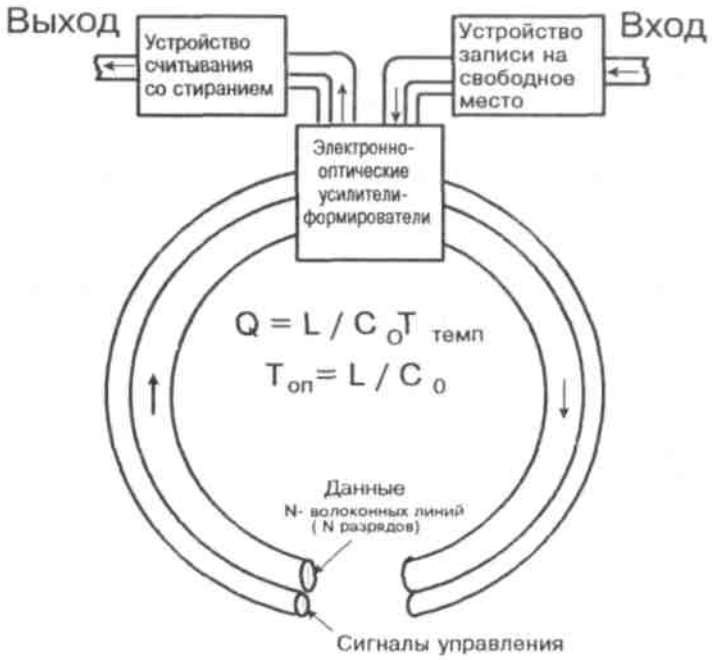


Рис.6. Динамическая память на оптоволокне. Q - *емкость памяти, L - длина оптоволокна, C - скорость распространения света в оптоволокне, Tr - скорость выполнения операций записи и считывания, Top - время выполнения операций записи и считывания.*

Приведенная на Рис.6, динамическая память представляет набор оптоволоконных линий задержек, замкнутых в кольцо. Количество линий может быть равно числу разрядов в слове памяти. По отдельному каналу передаются синхронизирующие импульсы, благодаря которым восстанавливается фаза импульсов в разрядных линиях задержки. Такая память аналогична памяти на дорожках диска, у которого линейная скорость перемещения носителя равна скорости света. Для работы этой памяти в системе потока данных достаточно реализовать запись информации по принципу на свободное место и считывание информации со стиранием. Работа таких схем записи и считывания показана на Рис.7, где представлена аналогичная память, реализованная полностью на оптических элементах и работающая в свободном пространстве.

В данном случае линией задержки является свободное пространство. Эта линия задержки замкнута сама на себя при помощи зеркал поляризационных оптических кубиков.

Единственным усиливающим элементом в этом зацикленном пространстве является вентиль несовпадения $ВН_1$. Если приходит импульс считывания, то $ВН_1$ закрывается и происходит стирание кадра данных, в котором может быть одно или несколько слов, и выдача этих данных через вентиль $В_1$.

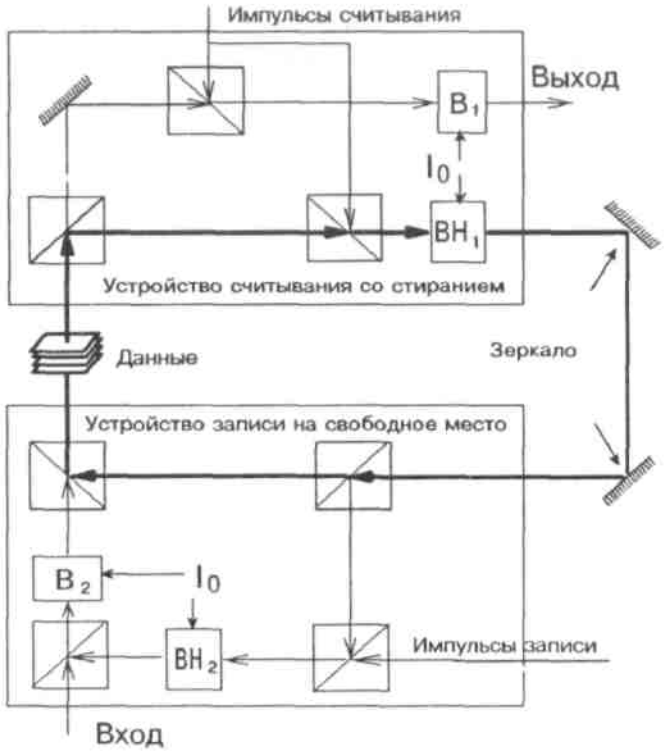


Рис.7. Оптическая динамическая память в свободном пространстве.
В - вентиль совпадения, ВН - вентиль несовпадения, 1_0 - синхроимпульсы.

В случае записи одновременно поступают на вход B_2 - данные и на BH_2 -импульс записи. Если место пустое, то на вход BH_2 приходит один импульс записи и BH_2 пропускает его на B_2, в результате чего через B_2 происходит запись кадра данных. Общая синхронизация схемы происходит за счет синхроимпульсов световой "накачки" I_0.

Из принципа работы динамической памяти вытекают следующие соотношения объема памяти Q, времен цикла $T_{оп}$ и темпа (скорости) ее работы $T_{темп}$:

$$Q = L / C_O T_{темп};\ T_{оп} = L / C_O \text{ или } Q = T_{оп} / T_{темп} \qquad (1)$$

Из приведенных соотношений видно, что динамические памяти будут иметь определенное преимущество перед полупроводниковыми по темпу записи и считывания, однако будут значительно проигрывать по циклу записи и считывания информации (времени операции).

В настоящее время проблема создания памяти в свободном пространстве соизмерима с созданием быстродействующего логического элемента. Лучшие работы в этом направлении, при использовании оптоволокна, приближаются к темпу 500 пс (2 ГГц).

Динамическая память на оптических принципах может найти применение в векторных исполнительных устройствах (Рис.4), где важен темп выдаваемой информации и не так остро стоит вопрос с временем доступа. В том случае, если время $T_{темп}$ динамической памяти приблизится к пикосекундному диапазону,

появится возможность ее использования в ассоциативной памяти и коммутаторе - распределителе. Однако, сравнительно высокие энергетические затраты делают такую память пока не конкурентоспособной.

2.3. Ассоциативная память

На Рис.8 приведена блок-схема памяти, полностью соответствующая функциям АП описанной выше суперЭВМ.

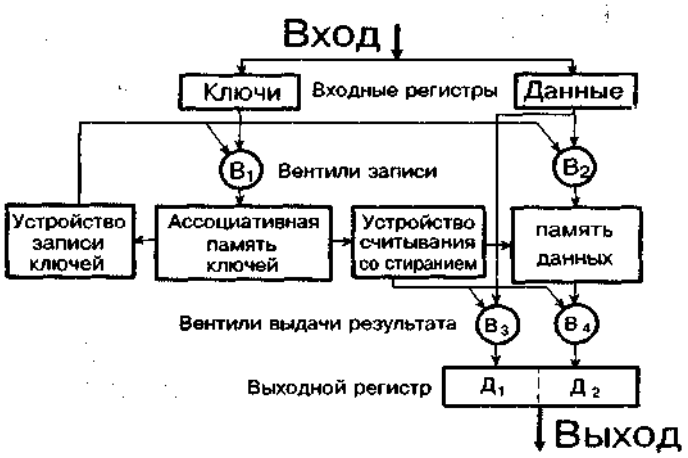


Рис.8. Блок-схема ассоциативной памяти.

На вход АП приходит данное с соответствующим ключом. Ключ в виде аргумента поступает на вход ассоциативной памяти ключей (АПК). При точном совпадении аргумента с одним из ключей ассоциативной памяти из нее выдается физический адрес соответствующего данного в памяти данных (ПД), по которому считываются необходимые данные и объединяются с данными искомого ключа. Одновременно со считыванием данных происходит их стирание в ПД и соответствующего ключа в АПК. Как правило, физический адрес АПК и ПД едины для одной и той же пары ключа и данных.

В том случае, если искомый ключ отсутствует в АПК, происходит запись этого ключа из входного регистра в АПК и соответствующего ему данного в ПД по одному и тому же физическому адресу. Типовая реализация этой блок-схемы на оптическом принципе приведена на Рис.9.

Искомый аргумент вместе с соответствующим данным отображается на управляющем транспаранте (УТ). Ассоциативной памяти ключей соответствует оптическое пространство А, а запоминающему устройству данных - оптическое пространство В. В соответствии с этим разбита и регистрирующая среда (РС). Шифратору физического адреса соответствует оптическая адресная система (АС). Фиксация результата ассоциативного поиска ключей производится матрицей фотоприемников ($МФП_1$), а регистрация считанных данных -матрицей фотоприемников ($МФП_2$). Хранимая в памяти информация регистрируется РС в виде прозрачных и непрозрачных участков, соответствующих записи двоичных "1" и "0". Таким же образом отображается информация на УТ.

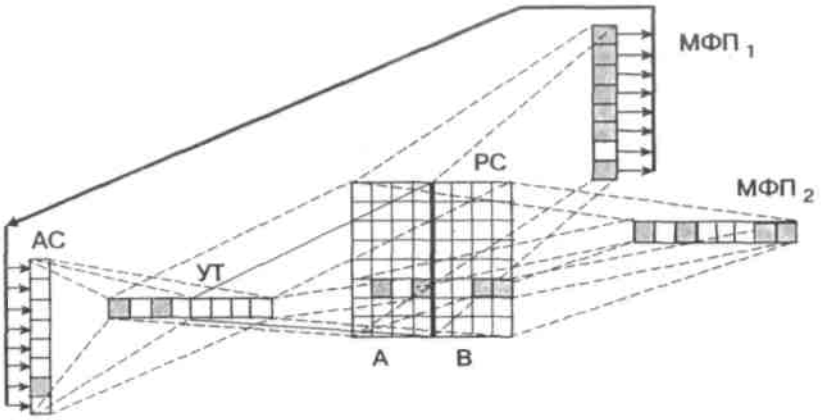


Рис.9. Ассоциативная память.

АС - адресная система, УТ - управляемый транспарант, РС - регистрирующая среда, $МФП_1$ - матрица фотоприемников для фиксации результата поиска, $МФП_2$ - матрица фотоприемников для данных, А - поле ключей, В - поле данных.

Для выполнения при ассоциативном поиске логических операций точного совпадения кодов двоичные разряды ключевых слов хранятся в памяти в парафазном коде:

$$\overline{b}_{1n} b_{1n} \overline{b}_{2n} b_{2n} ... \overline{b}_{nn} b_{nn} \quad ,$$

а предъявляемый аргумент поиска (ключ) кодируется светоклапанными элементами УТ инверсным парафразным кодом

$$\overline{a}_1 a_1, \overline{a}_2 a_2 ... \overline{a}_n a_n \quad .$$

Запись информации (признаков и связанных с ними данных) производится путем ее отображения на определенный участок РС с использованием оптической адресной системы (АС), функции которой может выполнять, например, электрически коммутируемая матрица полупроводниковых лазеров (МПЛ). При ассоциативном поиске АС высвечивает световые пучки, соответствующие всем М физическим адресам ячеек памяти, благодаря чему изображение инверсного парафазного кода аргумента поиска проецируется на все ячейки памяти ключей, и суммарные сигналы результатов оптического сравнения этого кода с парафазными кодами хранящихся признаков фиксируются $МФП_1$. Если один из ассоциативных признаков совпал с аргументом поиска, то на соответствующем элементе $МФП_1$ сигнал будет отсутствовать. Этот сигнал инвертируется и запускает соответствующий лазер АС, который производит считывание данных на $МФП_2$. В том случае, если искомый ключ не найден (все фотоприемники засвечены), происходит запись информации, находящейся на УТ в поле РС, для чего системой записи включается соответствующий лазер АС.

Из приведенной схемы работы АП видно, что наиболее принципиальной ее частью является операция ассоциативного поиска в памяти ключей (Рис. 10). В тоже время, именно здесь, в наибольшей степени могут использоваться характерные для оптики свойства широкополосности и широкоформатности передачи информации.

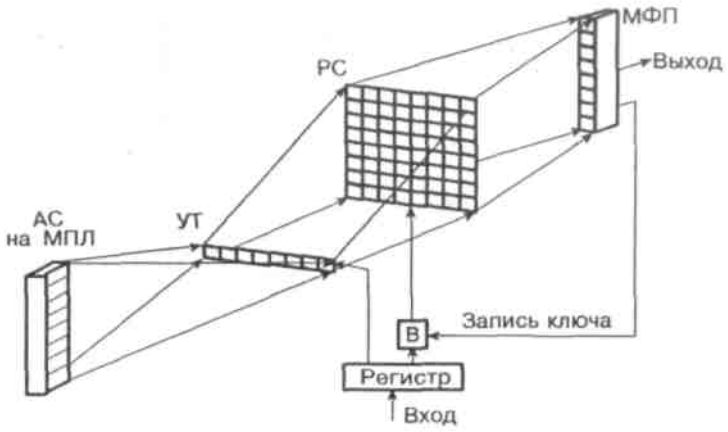


Рис.10. Ассоциативная память ключей.

АС - адресная система на матрице полупроводниковых лазеров, УТ - управляемый транспарант, РС - регистрирующая среда, МФП - матрица фотоприемников для фиксации результата поиска.

Фактически информация кода искомого ключа доставляется до каждого элемента ключевой памяти с сопоставлением всех разрядов каждого ключа в единой точке. Безусловно, в выполнении этих функций ассоциативной памяти оптика будет иметь преимущества перед электронными принципами обработки информации. В выполнении функций записи и считывания информации по физическому адресу, электронные принципы реализации этих функций на полупроводниковой базе в настоящее время будут иметь определенное преимущество и, в первую очередь, по энергетике, технологичности и физическим объемам реализации этих устройств.

Предельные параметры ассоциативной памяти (ее объем и производительность) всецело определяются соответствующими параметрами АПК. Основными параметрами ассоциативной АПК можно считать следующие: объем хранящихся в памяти ключей Q, количество разрядов ключа n, время поиска $t_П$ и темп обработки информации или производительность П, время записи ключа по физическому адресу $t_З$.

Для нормального функционирования скалярной части суперЭВМ новой архитектуры на первом этапе можно считать приемлемым объем памяти ключей $Q = 10^6$. Такая цифра может быть достаточной, имея в виду, что для векторного процессора за каждым ключом может содержаться от 10^2 до 10^4 слов в ПД.

Максимальная производительность такой АПК, исходя из заданной допустимой потребляемой мощности $P_{доп}$, будет определяться следующим соотношением [7]:

$$П_{max} = P_{доп}\,\eta / 2nWQ \qquad (2)$$

где: W - минимальная энергия срабатывания фотоприемника, необходимая для различения нуля и единицы, $\eta =$, где η_1 и η_2 - коэффициенты, учитывающие потери в лазере и оптической системе, соответственно.

При определении величины W необходимо исходить из требований надежности срабатывания фотоприемного устройства, выполняющего функции порогового

инвертора света. Воздействующий на него моноимпульсный сигнал от одномодового лазера с характерной для такого излучения пуассоновской статистикой фотонов должен содержать, по крайней мере, несколько тысяч фотонов [8], что в видимой области спектра эквивалентно энергии 10^{-15} Дж. Однако, в матрицах с большим числом фотоприемников их пороговая чувствительность определяется не столько статистическими свойствами светового сигнала, сколько факторами, связанными с неидентичностью и нестабильностью характеристик элементов, разбросом их параметров, кодозависимыми помехами как временного (предистория), так и пространственного характера (паразитные связи между соседними элементами). Вследствие этого, как показывает опыт практических разработок, реализовать в матрицах с большим числом фотоприемников пороговую чувствительность, превышающую $W = 10^{-14}$ Дж, представляется весьма проблематичной задачей. Практически η не превышает 10^{-2} ($\eta \le 10^{-1}$ и $\eta \le 10^{-1}$). Задаваясь предельно допустимой мощностью устройства $P_{доп} = 10^{4}$ Вт, ограниченной возможностью отвода тепла с поверхности устройства (при жидкостном охлаждении не превышает 20 Вт с кв.см.) и минимальным требуемым объемом памяти ключей $Q = 10^6$, $n = 50$ разрядам, получим предельную производительность АПК равную $П_{max} = 10^8$ оп/с.

Из соотношения (2) следует, что увеличение объема памяти в несколько раз, во столько же раз снижает и максимально возможную производительность АПК. Поэтому дальнейшее повышение производительности системы может быть достигнуто только путем расслоения памяти на параллельно работающие модули. Причем, если зафиксировать допустимую мощность на всю систему модулей, то увеличение производительности системы будет пропорционально количеству блоков при неизменной общей памяти Q. Если же зафиксировать предельную мощность на один модуль системы, то производительность системы будет квадратично возрастать от количества модулей N. Таким образом, если схемотехнически решить возможность модульной организации АПК, то можно существенно расширить возможности АП в части ее производительности и объема. Настоящий анализ проводился без учета технологических возможностей построения ассоциативной памяти ключей.

Более полный анализ дает возможность определить времена записи и считывания информации при работе с АПК. Темп считывания (поиска) в такой памяти, в основном, ограничен скоростью модуляции лазерных элементов и может достигать 1-0.1 нс. По этому параметру АПК, построенная на оптических принципах, будет иметь существенное преимущество перед полупроводниковой.

Гораздо хуже обстоит дело со временем записи, которое в настоящее время в РС, построенных на жидких кристаллах, не превосходит 1 мкс, в то время как АПК, построенная на базе полупроводниковой техники, может иметь время считывания и записи равным 10 нс. Поэтому, построение оптической АП возможно только с использованием групповой страничной записи ключей в память.

На Рис.11 представлена блок-схема оптической АП, обеспечивающей групповую запись ключей. Ключи, поступающие на такую систему, сначала проверяются на совпадение в небольшой ассоциативной полупроводниковой памяти, а затем в оптической.

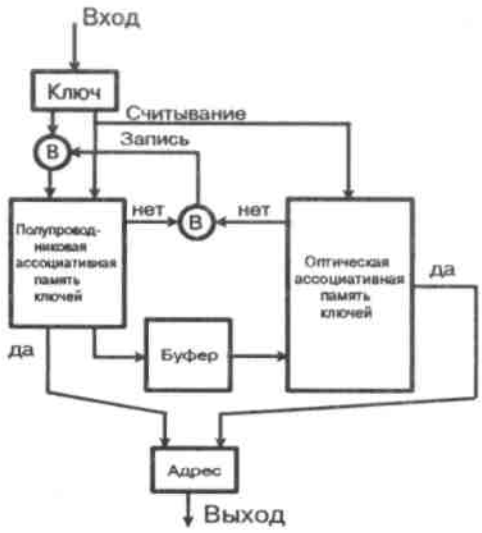


Рис.11. Блок-схема ассоциативной памяти ключей.

Если ключ найден в любой из двух памятей, он стирается и вырабатывается адрес. Если же ключа нет ни в той, ни в другой памяти, то он записывается сначала в полупроводниковую память, а затем передается в буфер для групповой записи в оптическую память. Использование метода групповой записи позволяет практически сравнять темп записи и считывания, но требует больших аппаратных затрат [5].

Интересными представляются работы по построению АП, в которых хранение информации осуществляется на полупроводниковых триггерах так же, как в обычной полупроводниковой памяти, причем каждый элемент памяти управляет оптическим элементом, меняющим свои свойства пропускания или отражения света.

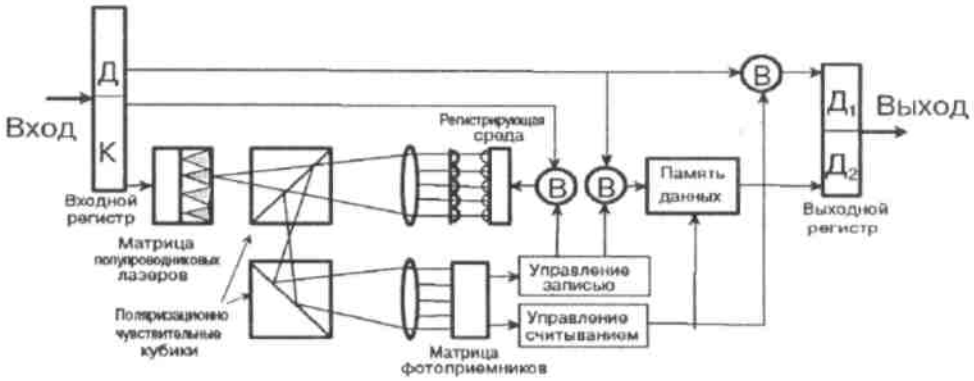


Рис.12. Оптоэлектронная ассоциативная память.

Развивая идею совместного использования электронных и оптических принципов обработки информации в одном устройстве уже на более совершенном, развивающемся в настоящее время технологическом уровне, можно говорить о перспективной АПК, основой которой будет двумерная матрица ячеек памяти ключей, состоящих из пространственно-временных модуляторов света, и размещенных в непосредственной близости от них и электрически

связанных с ними триггеров, хранящих информацию о кодах ключей. На Рис.12. приведена схема такого модуля ассоциативной памяти.

Перезапись ключей по адресу в плате памяти, шифрация и дешифрация физических адресов ячеек памяти, а также хранение связанной с ключами информации, осуществляется электронным устройством, имеющим в своем составе полупроводниковую память. Функции маскируемого регистра выполняет матрица полупроводниковых лазеров (МПЛ), на которой с помощью электронного дешифратора отображается инверсный код предъявляемого аргумента поиска. Физические адреса совпадений фиксируются матрицей фотоприемников. Оптическая система, состоящая из размещенных на фокусном расстоянии объективов, двойного растра линз, поляризационно-чувствительных кубиков и расщепителя световых пучков каждого из лазеров, на М световых пучков позволяет проецитировать в плоскость платы памяти микроизображения МПЛ и в плоскость МФП картину распределения s - компоненты излучения, возникающей за каждой из линз растра при отражении световых пучков от платы памяти.

Использование в качестве УТ полупроводниковой матрицы, позволяет в модуле АП реализовать схему одновременного поиска информации по многим ключам [11], что существенно расширяет пропускную способность модуля и АП в целом.

Таким образом, можно предполагать, что в недалеком будущем оптические и электронные принципы обработки информации, реализуемые на единой технологической базе, будут дополнять друг друга.

Дальнейшее развитие использования оптических принципов обработки информации в архитектуре суперЭВМ нового поколения связано с существенной интеграцией оптических и электронных принципов на единой технологической базе.

2.4. Коммутаторы

Определим требования к коммутаторам K_1 и K_2:

- K_1 должен обеспечивать передачу ста пакетов данных на все свободные исполнительные устройства за время 1 - 10 нс. Те данные, для которых не хватило исполнительных устройств, должны быть запомнены до следующего такта и т.д. Каждый пакет содержит 100 - 200 разрядов;
- K_2 должен обеспечивать передачу ста пакетов данных по ста направлениям в соответствии с адресом, имеющимся в каждом пакете, за время одного такта 1 - 10 нс;
- Кз должен обладать небольшим временем перекоммутации направлений -порядка нескольких десятков наносекунд, однако он должен передавать каждые 1 - 10 нс кванты информации объемом в страницу (128 х 64 бит) между всеми восемью направлениями одновременно по входу и выходу.

Пространственные коммутаторы

Пространственные коммутаторы бывают одноступенчатые, как правило матричные, и многоступенчатые. Типовой матричный коммутатор изображен на Рис.13. Информация с входных каналов при помощи оптических систем разводится на все j-ые элементы i-oro столбца. Каждый элемент имеет электронное

управление, которое делает каждый элемент либо прозрачным, либо отражающим и непропускающим свет. Информация с входных каналов при помощи оптических систем разводится на все j-ые элементы i-ого столбца. Каждый элемент имеет электронное управление, которое делает каждый элемент либо прозрачным, либо отражающим и непропускающим свет.

Оптические системы объединяют все i-ые элементы каждой j-ой строки в выходные каналы. Таким образом, каждый входной i-ый канал может быть скоммутирован с каждым выходным j-ым каналом в том случае, если во всех j-ых строках имеется не более одного элемента, пропускающего свет.

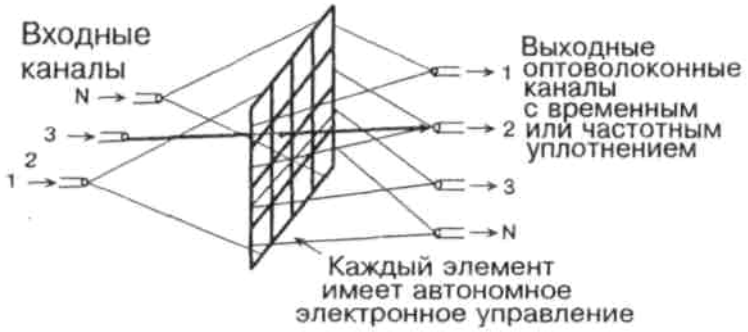


Рис.13. Матричный пространственный коммутатор.

По каналам может передаваться как последовательный код, так и параллельный (матрицами). В большинстве случаев передается последовательный код, а преобразование из параллельного в последовательный и обратно производится при входе в канал и, соответственно, при выходе из него. Таким образом, используются два основных преимущества оптики - широкополосность при уплотнении в один канал и широкоформатность при одновременной коммутации большого количества каналов. В последнее время стала возможной и интегральная реализация этой системы коммутации (Рис.14).

Рис.14. Интегральная реализация матричного коммутатора.

В этом случае в качестве разводящих и собирающих оптических систем могут быть использованы плоские голограммы в сочетании с пространственно-модулируемой матрицей света ПМС, которые могут быть созданы на жидких

кристаллах, на магнитооптике и на элементах, работающих на квантоворазмерных эффектах.

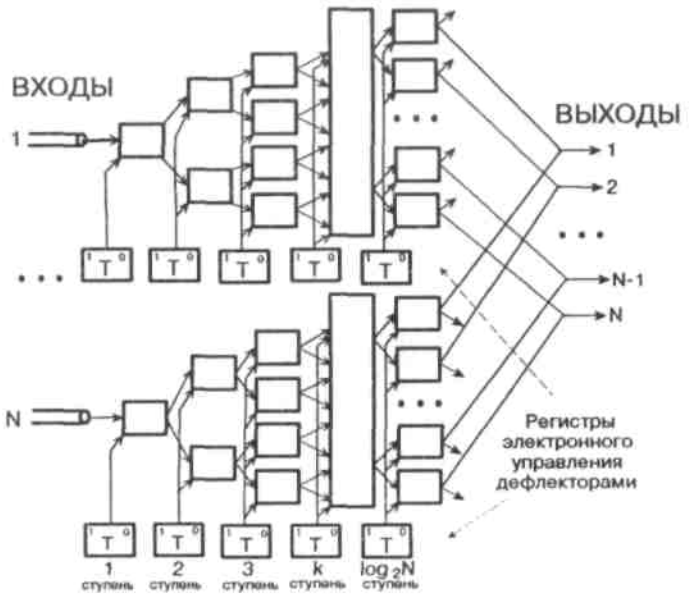


Рис.15. Многоступенчатый коммутатор на дефлекторах.

Пример многоступенчатого коммутатора света показан на Рис.15. Основным элементом такого коммутатора является дефлектор, управляемый электрическим сигналом. Дефлектор, в зависимости от поданного на него напряжения, меняет угол преломления и, таким образом, либо пропускает луч, либо отклоняет его. Выстраивая последовательно каскады дефлекторов, можно вести коммутацию луча по N каналов при $\log_2 N$ каскадов.

В настоящее время ведутся исследования возможности использования электрооптических дефлекторов на кристалле DKDP, акустооптических модуляторов света, интегрально-оптических ответвителей и т.д.

Многоступенчатые модуляторы будут иметь преимущества перед матричными при построении коммутационных систем тогда, когда будет выполняться соотношение $1 / N > \eta^{\log_2 N}$, где η - коэффициент полезного действия дефлектора, равный $\eta = E_{вых} / E_{вх}$, $E_{вых}$ и $E_{вх}$ - интенсивности светового сигнала на входе и выходе дефлектора.

К пространственным коммутаторам можно отнести и статический коммутатор с использованием линеек лазеров и фотоприемников. Его создание с использованием линеек лазеров, а в последствии возможно и матриц лазеров и фотоприемников, становится реальным благодаря успехам технологии последних лет в области интегральной оптики. Схема одноразрядного статического коммутатора приведена на Рис.16.

Каждый канал коммутации имеет лазерную линейку, работающую в достаточно легком режиме - допускается срабатывание только одного лазера из N находящихся в линейке. Лазеры управляются от дешифратора, выбирающего тот канал, с которым должна быть выполнена коммутация. Свет от возбужденного

лазера попадает в устройство статической коммутации и затем в фотоприемник выбранного канала линейки фотоприемников.

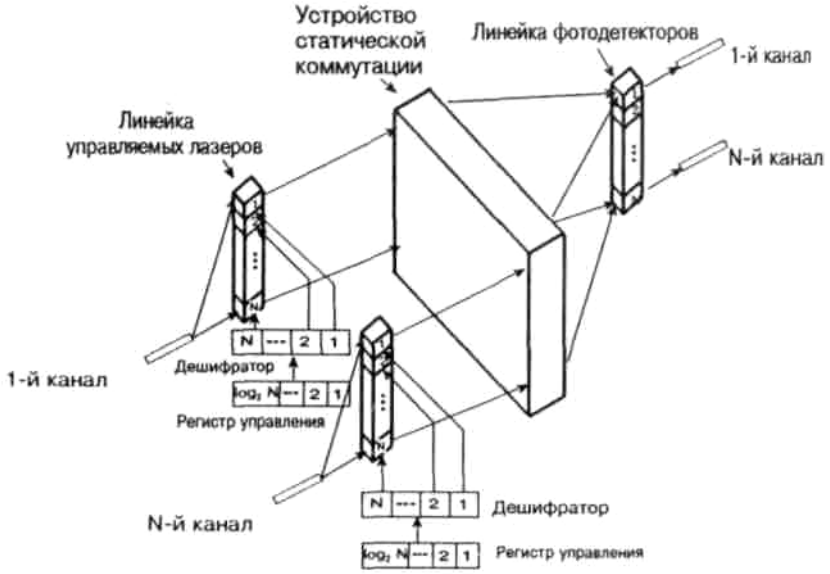


Рис.16. Одноразрядный статический коммутатор.

Устройство статической коммутации в данном случае выполняет функции объединения излучений одноименных лазеров всех N передающих направлений. Это устройство в зависимости от конкретных требований может быть выполнено как на базе средств геометрической оптики, так и с помощью объемных или плоских голограмм.

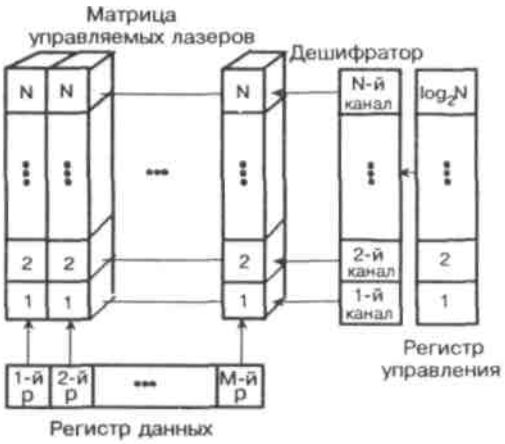


Рис.17. Одноканальный статический коммутатор.

Возможно, более простым решением будет использование оптоволоконных и оптических шайб. Если на передающей стороне иметь матрицу, состоящую из m линеек лазеров, а на приемной стороне соответствующую матрицу фотоприемников, то можно за один цикл работы коммутатора передавать m- разрядное

слово. В этом случае матрица лазеров, наряду с управлением по столбцам от дешифратора каналов, должна управляться по строкам от разрядов регистра передаваемого данного. Схема одноканального m-разрядного коммутатора приведена на Рис.17.

Частотный принцип коммутации сигналов.

Схема коммутации, представленная на Рис.18, базируется на управляемом по частоте и по запуску лазере.

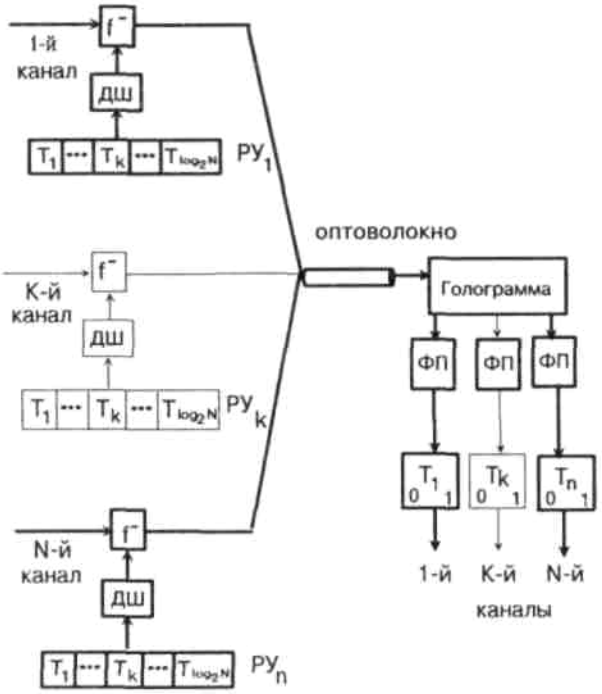


Рис.18. Коммутатор, использующий частотное разделение каналов.

Предполагается, что лазер запускается оптическим сигналом, выходящим из оптоволокна входного канала. Необходимая коммутация входного канала задается кодом входного канала, указанным на регистре управления (РУк). В зависимости от сигнала, выходящего со схемы дешифратора ДШ, задается частота генерации лазера.

Выходящие с лазеров импульсные последовательности, "окрашенные" различными частотами, объединяются в одном волокне, либо пучком волокон передаются на одну или N голограмм, которые разделяют информацию по частоте и направляют на соответствующие фотоприемники ФП выходных каналов.

Разработка управляемых по частоте полупроводниковых лазеров в настоящее время ведется в ряде физических институтов РАН.

2.5. Мультиплексоры и демультиплексоры

Описанные выше схемы коммутаторов предполагают предварительное сжатие информации в канале, используя широкополосные возможности оптики. Другими словами, там, где полупроводниковая техника для обеспечения пропускной

способности информации вынуждена работать параллельным кодом, оптика имеет возможность работать последовательным кодом, существенно упрощая проблемы коммутации данных в суперЭВМ.

Мультиплексоры и демультиплексоры работают как на принципе временного уплотнения канала, так и на принципе частотного уплотнения. Пример временного принципа уплотнения канала показан на схеме Рис.19.

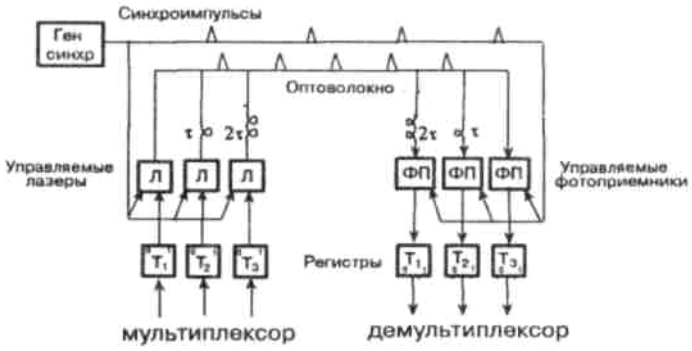


Рис.19. Мультиплексор и демультиплексор с временным уплотнением.

Уплотнение сигнала происходит за счет разной временной задержки в оптоволокне на передающей стороне и восстановления этой задержки в обратном порядке на приемной стороне. На передающей стороне, перед выдачей сигнала в оптоволокно, задержка с каждым разрядом увеличивается на т, а на приемной стороне задержка сигнала с каждым разрядом уменьшается на т. При этом задержка сигнала на первом разряде на приемной стороне соответствует задержке на последнем разряде передающей стороны. Синхронизирующие импульсы запускают лазеры, управляемые от кодовых разрядов и те же синхронизирующие импульсы стробируют в фотоприемниках приходящую световую информацию Фактически, лазеры выполняют функцию выходных вентилей, а фотоприемники - входных вентилей на приемном конце. Выдаваемые с каждого выходного лазера световые импульсы объединяются и передаются по оптоволокну на приемный конец, где равномерно разводятся по всем фотоприемникам.

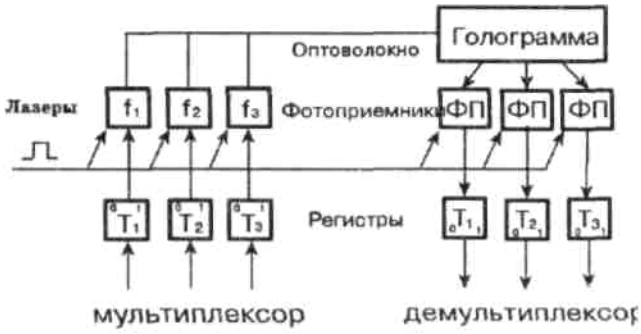


Рис.20. Мультиплексор и демультиплексор с частотным уплотнением.

Работа мультиплексора и демультиплексора на принципе частотного уплотнения (Рис.20) ничем не отличается от работы коммутатора на том же принципе. Разница

состоит в том, что нет необходимости управлять лазером по частоте. Каждый разряд кода управляет лазером, фиксированно настроенного на частоту. Управляющий сигнал открывает лазеры на приемном конце и те из них, триггера которых стоят в положении "1", запускаются. Импульсы со всех разрядов собираются в единую оптоволоконную линию, и на приемной стороне, посредством голограммы, дешифрируются по частоте и выдаются на фотоприемники соответствующих разрядов.

Работы по созданию уплотненных каналов передачи данных можно считать наиболее продвинутыми в настоящее время.

2.6. Тасователи

Тасователь является распределительным устройством, предназначенным для перестановки элементов вектора в соответствии с вектором индексов (вектором, каждый элемент которого содержит новый индекс соответствующего элемента обрабатываемого вектора).

В оперативной памяти векторного исполнительного устройства (памяти векторов) расположены вектора размером от 1 до 128 страниц каждый (размер страницы - 128 слов). Возможно не полное заполнение страниц элементами. Тасователь может выполнять операции как над отдельными страницами, так и над векторами в целом. Операция распределения над вектором выполняется постранично.

Индекс элемента вектора состоит из номера страницы и номера элемента в странице. Номер элемента в странице определяет физический номер модуля оперативной памяти; номер страницы в векторе определяет физический адрес внутри модуля оперативной памяти. Программно-аппаратными средствами обеспечивается постраничный вызов данных из оперативной памяти в соответствии с их виртуальными номерами.

Таким образом, на вход тасователя каждый такт подается страница элементов упорядочиваемого вектора вместе со страницей соответствующих индексов, определяющих новое место каждого элемента в странице ($i_{НОВ}$) или векторе ($i_{НОВ}\,j_{НОВ}$). Отсутствие индекса соответствует блокировке данного элемента (маскирование).

Возможны следующие операции:

- переставить местами элементы страницы вектора в соответствии со страницей новых индексов;
- переставить местами элементы страницы вектора в соответствии с новыми индексами другого вектора;
- наложить маску на элементы вектора;
- совмещение этих операций;
- наложить одну страницу на другую с преобладанием поля одной из страниц.

Если для выполнения операций над отдельными страницами векторов достаточно использовать простейший коммутатор векторно-матричного типа, то для выполнения тех же операций над векторами из многих страниц необходима достаточно сложная схема устройства (Рис.21).

Прежде всего необходимо реализовать устройство, на котором можно было бы достаточно легко выполнить наиболее типовую операцию над матрицей: операцию транспонирования матрицы (то есть преобразования вектора, представляющего

собой выстроенную последовательность столбцов матрицы в вектор, представляющий собой выстроенную последовательность строк этой же матрицы).

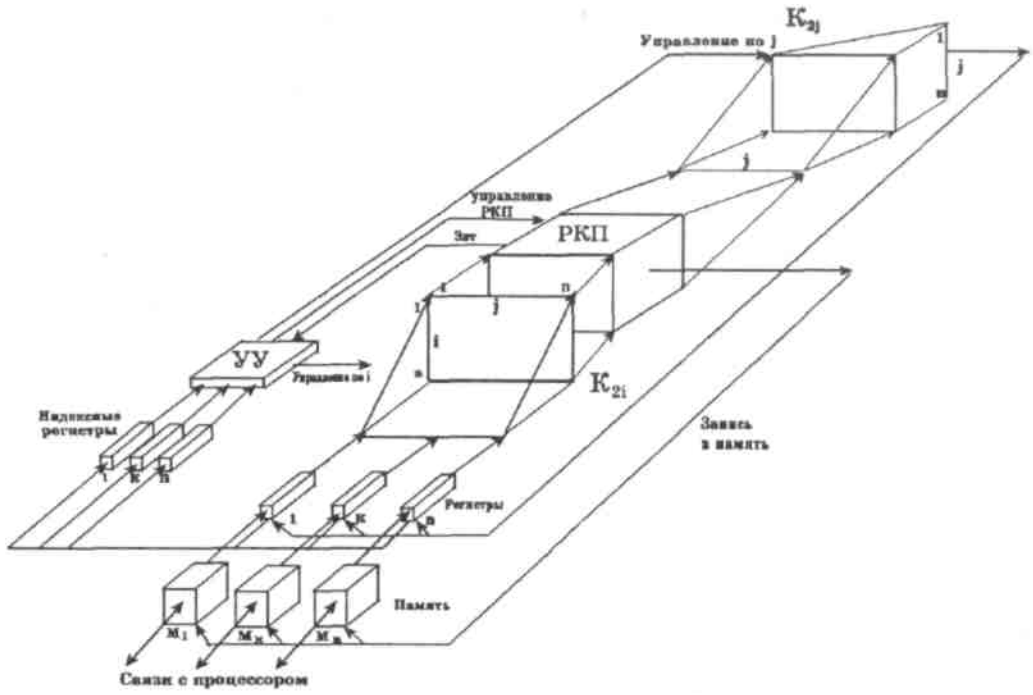


Рис.21. Тасователь.

Простейшим способом эта перестановка может быть выполнена на относительно небольшой многопортовой регистровой памяти. Для матрицы размерностью n x n такая память может состоять их n параллельно работающих кристаллов, содержащих по n регистров каждый. В каждый кристалл осуществляется последовательная поразрядная запись одновременно n чисел, считанных из векторной оперативной памяти и принадлежащих к одной строке вектора (далее предполагается, что страница вектора совпадает со строкой матрицы).

Считывание информации с этого кристалла производится пословно в соответствии с адресом слова. За n тактов работы векторной оперативной памяти могут быть последовательно заполнены n таких кристаллов с 1-го по n-й. Считывание со всех n кристаллов производится одновременно пословно параллельным кодом и получившаяся строка записывается обратно в оперативную память таким образом, что слово, считанное из первого кристалла, пишется в оперативную память первого элемента.

Если страницы вектора, представляющие строки старой матрицы, считать из оперативной памяти последовательно в соответствии с виртуальным номером страницы и записывать последовательно в кристаллы с 1-го по n-й, то считанный с кристаллов в оперативную память вектор будет представлять собой матрицу, у которой строки поменяются местами со столбцами.

Поэтому, как основное звено устройства распределения вектора, должен функционировать вышеописанный регистровый куб памяти с соответствующим

правлением (Рис.22). В этом кубе фактически формируется результирующий вектор, который затем считывается в оперативную память.

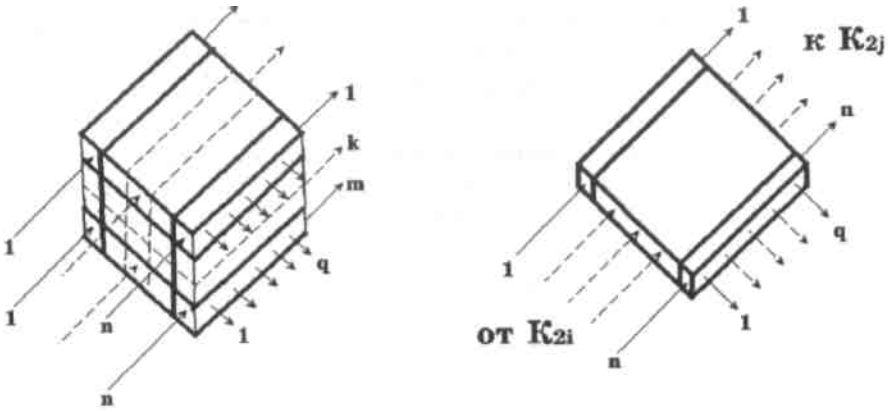


Рис.22. Регистровый куб памяти.

Решая задачу перестановки элементов вектора в общем виде, мы должны иметь возможность переместить каждый элемент исходного вектора на некоторую новую позицию в соответствии с вектором индексов. Имея в виду постраничное хранение элементов вектора в оперативной памяти, попытаемся выполнить эту задачу постранично и поэтапно с использованием регистрового куба памяти. Для этого произведем следующую последовательность действий:

- считаем из оперативной памяти страницу исходного вектора и соответствующую ей страницу вектора новых индексов;
- посредством коммутатора K_{2i} направим каждый элемент страницы в соответствующий регистр куба памяти по индексу i;
- после записи всех страниц регистра в регистровый куб, передаем каждую i-ю страницу с регистрового куба на коммутатор K_{2j} и упорядочиваем все ее элементы в соответствии с новым значением j. После этого через коммутатор K_{2i} записываем страницу в регистровый куб;
- после того, как обработаны все страницы вектора, производим постраничную запись их из регистрового куба параллельным кодом в оперативную память.

Если не принять специальных мер, работа такой схемы будет наталкиваться на конфликтные ситуации, заключающиеся в том, что через одно и то же j-е направление может быть заблокировано перемещение нескольких элементов вектора. Эти конфликты можно снять, сделав двойными регистры куба (Рис.23).

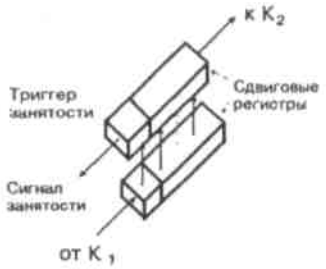


Рис.23. Регистр куба.

После того, как информация из $К_{2i}$ поступает в регистр куба, она передается в спаренный с ним регистр, выдающий информацию на $К_{2j}$. В том случае, если спаренный регистр занят, возникает прерывание и запускается работа второго коммутатора по всей этой странице регистрового куба. После работы $К_{2j}$, информация записывается через $К_{2i}$ обратно в ту же строку регистрового куба.

Можно оценить время работы тасователя по основным операциям над матрицами. Так, операция транспонирования матрицы размерностью n займет время, равное $t_1 = 2nT_{ОП} + nT_{K2i}$, где $T_{ОП}$ - время одного цикла оперативной памяти, а T_{K2i} - время работы коммутатора $К_{2i}$ вместе с временем записи данных в регистровый куб.

Перестановка элементов вектора в соответствии с вектором индексов в самом общем случае займет время, равное:

$$t_2 = 3nT_{ОП} + n(T_{K2i} + T_{K2j}) + 2X(T_{K2i} + T_{K2j}), \qquad (3)$$

где T_{K2i} - время работы коммутатора $К_{2i}$, а X - число конфликтных ситуаций, возникших при перестановке.

Необходимо отметить, что традиционно программно-аппаратными средствами задача перестановки элементов матрицы в соответствии с заданными индексами, займет время более, чем:

$$3n^2 (T_{оп} + T_{опер}),$$

где $T_{опер}$ - время выполнения короткой операции на процессоре.

Таким образом, тасователь позволяет (при n = 100) на два порядка сократить время выполнения этой операции, широко распространенной при работе с векторами, и значительно снизить количество передач данных между ассоциативной памятью и оперативной памятью векторного исполнительного устройства. Предполагается, что эта векторная операция довольно быстро выполняется на скалярном процессоре с ассоциативной памятью.

Заключение

Суммируя результаты анализа, можно построить таблицу, показывающую использование оптических средств обработки информации в различных устройствах суперЭВМ нового типа.

Таблица

У с т р о й с т в а	**В настоящее время**	**Через 10 - 20** лет
Исполнительные устройства (логические вентили)	Полупроводники	Оптика?
Оперативная память	Полупроводники	Оптика?
Ассоциативная память	Полупроводники или оптика	Оптика
Коммутаторы	Оптика или полупроводники	Оптика

На основании проведенного выше анализа построена детальная блок-схема суперЭВМ (Рис.24), позволяющая оценить возможности применения оптических принципов обработки информации в архитектуре суперЭВМ.

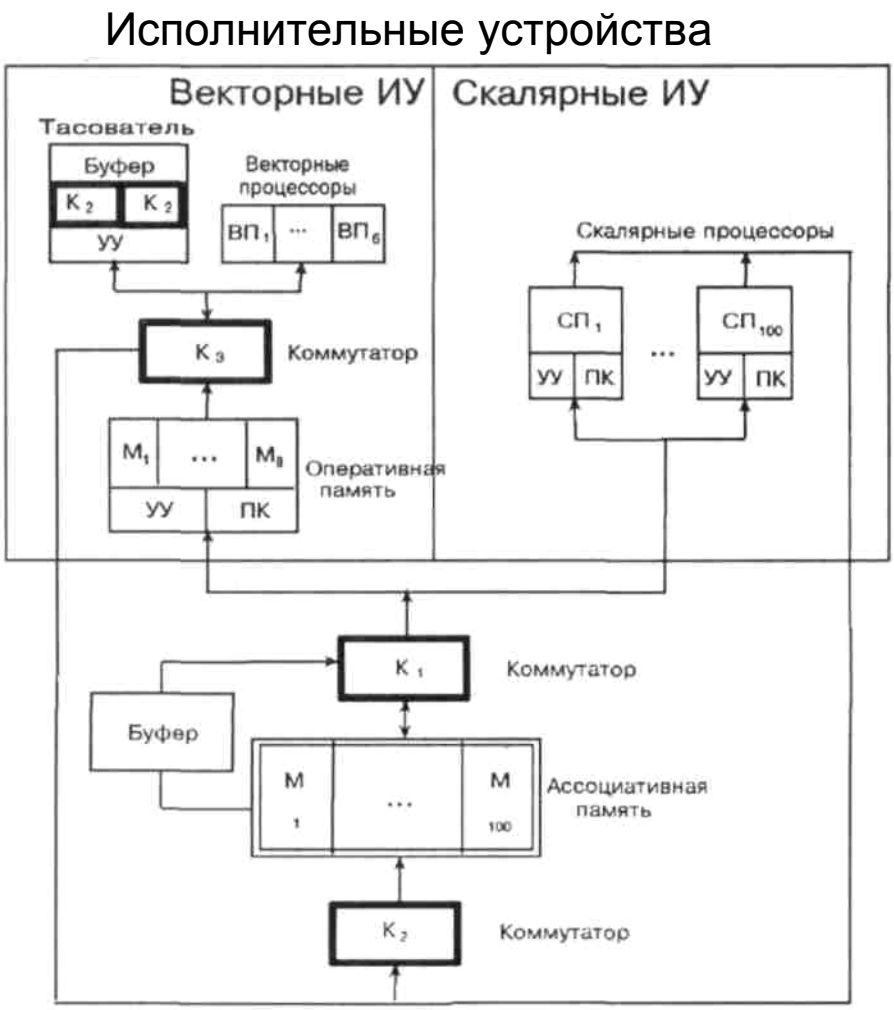


Рис.24. Схема суперЭВМ нового типа.

Устройства, показанные на схеме квадратами с жирными линиями, могут быть уже в настоящее время выполнены с использованием оптоэлектронных принципов. Оптическая ассоциативная память, показанная на схеме квадратом с двойными линиями, предпочтительна полупроводниковой памяти.

Как видно из Таблицы и Рис.24 процент оборудования, необходимого для построения суперЭВМ с нетрадиционной архитектурой, выполненного с использованием оптических элементов может достигать 30 - 60%.

Литература

1. В.С. Бурцев. Анализ результатов испытаний МВК "Эльбрус-2" и дальнейшие пути его развития. - М., 1988. (Препринт ОВМ АН СССР N 208).
2. В.С.Бурцев, Е.А.Кривошеев, В.Д.Асриэли, П.В.Борисов, К.Я.Трегубов. Векторный процессор МВК "Эльбрус-2". СуперЭВМ. 1989. (Сб. научных трудов ОВМ АН СССР).
3. В.С.Бурцев. Тенденции развития суперЭВМ. Оптические принципы обработки информации в архитектуре суперЭВМ. М.,1992. (Препринт ВЦКП РАН N 24).
4. V.S.Burtsev, V.B.Fyodorov. Associative memory of new generation super-computers based on optical information processing principles. Holography and Optical Information Processing, 1991, V. 1731, p. 201-216.
5. Проект ОСВМ. 1993.
6. В.С.Бурцев. Тенденции развития суперЭВМ. Вычислительные машины с нетрадиционной архитектурой суперЭВМ. М.: Наука, 1990. Сб. 1. С.3-26.
7. V.S.Burtsev, V.B.Fyodorov. Associative memory of new generation Supercomputers based on optical information processing principles. Holography and optical information processing. 1991, v1731, P.201-206.
8. G.Popadopoulos, D.Culler. Monsoon: an Explicit Token-Store Architecture, Sigarch Computer Architecture News vol.18, no.2, 1990.
9. D.Culler. The Explicit Token Store, Journal of Parallel and Distributed Computing, vol.10, 289-308, 1990.
10. G.Popadopoulos, K.Traub. Multithreading: A Revisionist View of Dataflow Architectures. Sigarch Computer Arch. News, vol.19, no.3, 1991.

Ассоциативная память на принципах оптической обработки информации для суперЭВМ нового поколения

В.С.Бурцев, В.Б.Федоров

Аннотация

Исследовании предельные возможности построения ассоциативной памяти (АП) по ее объему и быстродействию. Предложены различные структурные возможности увеличения быстродействия АП по записи информации. Приведены примеры схемных решений увеличения объема и пропускной способности АП за счет модульного построения. Приведена схема, позволяющая построить модуль АП на базе полупроводниковых интегральных схем с использованием оптоэлектронных принципов.

1. Предельные параметры ассоциативной памяти

На Рис.1 представлена блок-схема памяти, разработанная для новой нетрадиционной архитектуры суперЭВМ.

На вход ассоциативной памяти приходит данное с соответствующим ключом. Ключ в виде аргумента поступает на вход ассоциативной памяти ключей (АПК). В случае точного совпадения ключа с одним из ключей ассоциативной памяти из нее выдается физический адрес соответствующего данного в память данных (ПД), по которому считываются необходимые данные и объединяются с данными искомого ключа. Одновременно со считыванием данных происходит их стирание в ПД и стирание соответствующего ключа в АПК. Как правило, физический адрес АПК и ПД едины для одной и той же пары ключа и данных.

В том случае, если искомый ключ отсутствует в АПК, происходит запись этого ключа из входного регистра в АПК и соответствующего ему данного в память ПД по одному и тому же физическому адресу. Типовая реализация этой блок-схемы на оптическом принципе приведена на Рис.2.

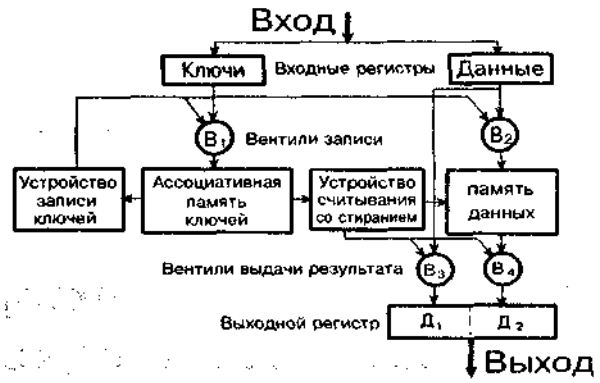


Рис.1. Блок-схема ассоциативной памяти.

Искомый аргумент вместе с соответствующим данным отображается на управляющем транспаранте (УТ). Ассоциативной памяти ключей соответствует оптическое пространство А, а памяти данных оптическое пространство В. В соответствии с этим разбита и регистрирующая среда (РС). Шифратору физического адреса соответствует оптическая адресная система (АС). Фиксация результата ассоциативного поиска ключей производится матрицей фотоприемников ($МФП_1$), а регистрация считанных данных матрицей фотоприемников ($МФП_2$).

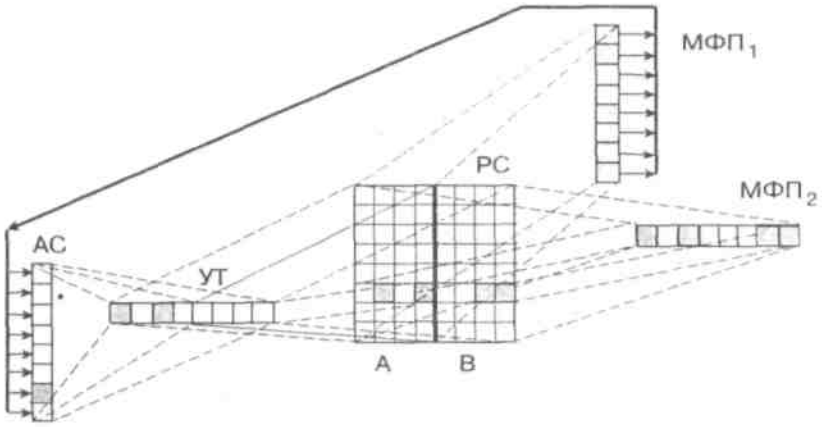


Рис.2. Ассоциативная память.

Хранимая в памяти информация регистрируется РС в виде прозрачных и непрозрачных участков, соответствующих записи двоичных "1" и "0". Таким же образом отображается информация на УТ.

Для выполнения при ассоциативном поиске логических операций точного совпадения кодов двоичные разряды ключевых слов хранятся в памяти в парафазном коде b_{1n} b_{2n} $b_{2n}b_{nn}...b_{nn}$, а предъявляемый аргумент поиска (ключ) кодируется светоклапанными элементами УТ инверсным парафразным кодом a_1 a_1 a_2 a_2 ... a_n a_n.

Запись информации (признаков и связанных с ними данных) производится путем передачи изображения на определенный участок РС с использованием оптической адресной системы, функции которой может выполнять, например, электрически коммутируемая матрица полупроводниковых лазеров (МПЛ). При ассоциативном поиске АС высвечивает световые пучки, соответствующие всем М физическим адресам ячеек памяти, благодаря чему изображение инверсного парафазного кода аргумента поиска проецируется на все ячейки памяти ключей, и суммарные сигналы результатов оптического сравнения этого кода с парафазными кодами хранящихся признаков фиксируются $МФП_1$. Если один из ассоциативных признаков совпал с аргументом поиска, то на соответствующем элементе $МФП_1$ сигнал будет отсутствбвать. Этот сигнал инвертируется и запускает соответствующий лазер АС, который производит считывание данных на $МФП_2$. В том случае, если искомый ключ не найден (все фотоприемники засвечены), происходит запись информации, находящейся на УТ в поле РС, для чего системой записи включается соответствующий лазер АС.

Из приведенной схемы работы АП видно, что наиболее принципиальной ее частью является операция ассоциативного поиска в памяти ключей (Рис.3). В тоже время именно здесь, в наибольшей степени могут использоваться характерные для оптики свойства широкополосности и широкоформатности передачи информации.

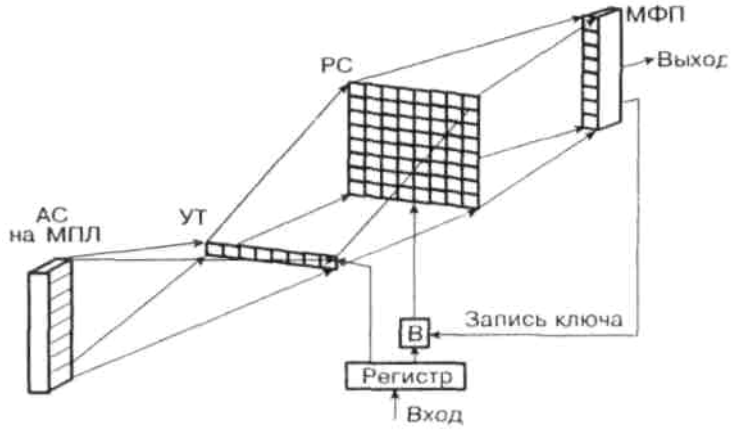


Рис.3. Ассоциативная память ключей. *АС - адресная система на матрице полупроводниковых лазеров: УТ- управляемый транспарант; РС - регистрационная среда; МФП - матрица фотоприемников фиксации результата поиска; В - вентиль записи.*

Фактически информация кода искомого ключа доставляется к каждому элементу ключевой памяти с сопоставлением всех разрядов каждого ключа в единой точке. Безусловно, в выполнении этих функций ассоциативной памяти оптика будет иметь преимущества перед электронными принципами обработки информации. При выполнении функций записи и считывания информации по физическому адресу, электронные принципы реализации этих функций на полупроводниковой базе в настоящее время имеют определенное преимущество

и, в первую очередь, по энергетике, технологичности и физическим объемам реализации этих устройств.

Предельные параметры ассоциативной памяти (ее объем и производительность) всецело определяются соответствующими параметрами АПК. Основными параметрами ассоциативной АПК можно считать следующие: объем хранящихся в памяти ключей Q, количество разрядов ключа n, время поиска $t_п$ и темп обработки информации или производительность П, время записи ключа по физическому адресу $t_з$

Для нормального функционирования скалярной части суперЭВМ новой архитектуры на первом этапе можно считать приемлемым объем памяти ключей $Q = 10^6$. Такая цифра может быть достаточной, при условии, что для векторного процессора за каждым ключом может содержаться от 10^2 до 10^4 слов в ПД.

Максимальная производительность такой АПК при заданной допустимой потребляемой мощности Рдоп, будет определяться следующим соотношением [7]:

$$П_{max} = P_{доп}\,\eta / 2nWQ \qquad (1)$$

где W - минимальная энергия срабатывания фотоприемника, необходимая для различения нуля и единицы, $\eta = \eta_1\eta_2$, где η_1 и η_2 коэффициенты, учитывающие потери в лазере и оптической системе, соответственно.

При определении величины W необходимо исходить из требований надежности срабатывания фотоприемного устройства, выполняющего функции порогового инвертора света. Воздействующий на него моноимпульсный сигнал от одномодового лазера с характерной для такого излучения пуассоновской статистикой фотонов должен содержать, по крайней мере, несколько тысяч фотонов [8], что в видимой области спектра эквивалентно энергии 10^{-15} Дж. Однако, в матрицах с большим числом фотоприемников их пороговая чувствительность определяется не столько статистическими свойствами светового сигнала, сколько факторами, связанными с неидентичностью и нестабильностью характеристик элементов, разбросом их параметров, кодозависимыми помехами как временного (предыстория), так и пространственного характера (паразитные связи между соседними элементами). Вследствие этого, как показывает опыт практических разработок, реализовать в матрицах с большим числом фотоприемников пороговую чувствительность, превышающую $W=10^{-14}$ Дж, представляется весьма проблематичной задачей. Практически g не превышает 10^{-2} ($\eta \le 10^{-1}$ и $\eta \le 10^{-1}$). Задаваясь предельно допустимой мощностью устройства $P_{доп} = 10^4$ Вт и минимальным требуемым объемом памяти $Q = 10^6$ ключей, n = 50 разрядам, получим предельную производительность АПК равную $П_{max} = 10^8$ оп/с. Предельно допустимая мощность ограничена возможностью отвода тепла с поверхности устройства, которая в случае жидкостного охлаждения не превышает 20 Вт с квадратного сантиметра.

Из соотношения (1) следует, что увеличение объема памяти в несколько раз, во столько же раз снижает и максимально возможную производительность АПК. Поэтому дальнейшее повышение производительности системы может быть достигнуто только путем расслоения памяти на параллельно работающие модули. Причем, если зафиксировать допустимую мощность на всю

систему модулей, то увеличение производительности системы будет пропорционально количеству блоков при неизменной общей памяти Q. Если же зафиксировать предельную мощность на один модуль системы, то производительность системы будет квадратично возрастать в зависимости от количества модулей N. Таким образом, если схемотехнически решить задачу модульной организации АПК, то можно существенно расширить возможности АП в части ее производительности и объема.

Настоящий анализ проводился без учета технологических возможностей построения ассоциативной памяти ключей.

На Рис.4. приведена рабочая схема АП, функционирующей в полном соответствии со схемой Рис.2.

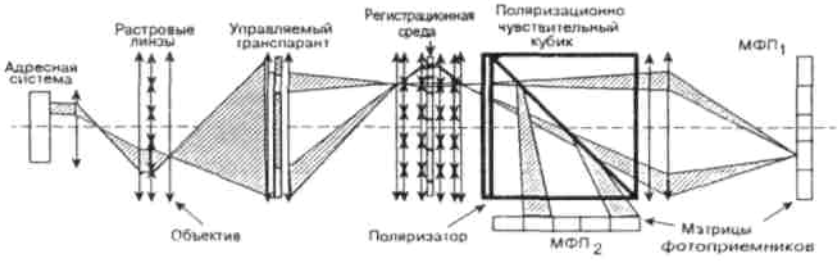


Рис.4. Рабочая схема АП.

В качестве светочувствительной РС используется среда, позволяющая осуществлять позитивную запись микрокадров (с просветлением участков, в которые записываются "I") уменьшенного изображения УТ, допускающая стирание и повторную запись микроизображений.

Два матричных фотоприемника предназначаются для считывания хранящейся в микрокадрах информации ($МФП_1$ с числом элементов 2n, равным количеству светоклапанов в УТ) и фиксации физических адресов, ключевые слова которых совпали с аргументом поиска (МФП с числом светочувствительных элементов Q, равным количеству микроизображений). В зависимости от того, в каком режиме адресном или ассоциативном работает память, световые пучки, прошедшие через РС, направляются на соответствующий МФП. Для этой цели используется интерференционный поляризационно - чувствительный кубик (ПЧК) и установленный перед ним модулятор плоскости поляризации (МП).

В адресном режиме АП работает так же как, например, растровая фотоскопическая память. При считывании информации все светоклапанные ячейки управляемого транспаранта переводятся в режим пропускания света, освещается микрокадр, соответствующий заданному адресу, и его изображение, проецируемое растровыми линзами и коллективным объективом в плоскость размещения $МФП_2$, регистрируется этим матричным фотоприемником. При ассоциативной обработке информации устройство работает как это было описано для общей схемы на Рис.2.

Рассмотрим конструкторско - технологические ограничения, возникающие при реализации ассоциативной памяти, функционирующей в соответствии с рабочей схемой, приведенной на Рис.4.

Объем информации АПК в битах (N) определяется площадью регистрирующей среды S и плотностью хранения в ней информации ρ и может быть оценен следующим соотношением:

$$N = \rho S = S(\lambda\chi F/D)^{-2}, \qquad (2)$$

где D и F- соответственно диаметр и фокусное расстояние объектива, в фокальной плоскости которого размещена РС; χ - константа порядка нескольких единиц, учитывающая выбранный критерий разрешения световых пятен (пикселов) и распределение интенсивности в поперечном сечении фокусируемого светового пучка. Если в качестве источника излучения применяется одномодовый лазер с гауссовым распределением интенсивности, то при χ = 2,9 в дифракционно ограниченных оптических системах с круговыми апертурными диафрагмами при оптимальном радиусе гауссова пучка можно получить концентрацию энергии в световых пятнах, превышающую 95%. Тем самым практически исключаются взаимные помехи между соседними пикселами в формируемом в поле РС изображении [10].

В качестве примера определим максимальный объем АПК со следующими параметрами оптической системы: F/D = 4, S = 10^2 см, λ = 0,85 мкм. На основании (2) найдем $N_{max} = 10^7$ бит при диаметре пикселов 10 мкм. Для более светосильной оптики и большей площади РС величина N_{max} может быть большей. Однако, поскольку с увеличением диаметра и светосилы объективов резко возрастают различного рода аберрации, снижающие плотность хранения информации q при побитовой форме ее регистрации, то получаемые из выражения (2) значения информационной емкости АПК будут в большей степени отличаться (в сторону завышения) от достижимых в реальных оптических системах. Поэтому в практических разработках величину $N_{max} = 10^7$ бит при S = 10^6 бит/ см или Q = 10^5 ключей можно считать предельно возможной.

Теперь оценим предельную производительность модуля АПК при максимальном ее объеме. В нашем случае, учитывая возможность конвейерной организации работы АПК, ее производительность будет определяться временем $t_п$, которое измеряется интервалом времени с момента завершения отображения кода аргумента поиска на УТ до получения результата ассоциативного опроса всех ячеек в модуле памяти ключей. Это время, в свою очередь, складывается из времени задержки оптического сигнала t_o , возникающей на пути его распространения от плоскости УТ до входной плоскости $МФП_1$ и времени срабатывания элементов $t_э$ фиксации совпадений.

Для традиционной оптической схемы АПК, выполненной на основе проекционной оптики (Рис.4), величина to может быть оценена как

$$t_o \approx 3\ F/c, \qquad (3)$$

где с - скорость света. При значениях параметров оптической системы, которые использовались для оценки N_{max} (Q = 10^5 слов), величина t_o = 1,5 нс, а при Q = 10^4 слов t_o = 0,5 нс.

Время же tэ может быть определено следующим соотношением:

$$t_э = W_1 / P_б , \quad (4)$$

где $P_б$ - световая мощность, приходящаяся на один бит (пиксел) в матрице фотоприемников при поиске по ключу, W_1 - минимальная световая энергия достаточная для регистрации факта несовпадения по одному биту поиска. Величина $P_б$ - может быть определена двумя различными соотношениями: через световую мощность $P_{сл}$, которую необходимо выделять одному лазеру матрицы АС для адресации к одному слову

$$P_б = P_{сл}\, \eta/2n, \quad (5)$$

и через выделяемую тепловую мощность на единицу площади в матрице фотоприемников

$$P_б = 2P_T / S, \quad (6)$$

где $\eta_2 = \eta_м\, \eta_т\, \eta_с\, \eta_р$ коэффициент, учитывающий световые потери в оптической системе на участках между АС и РС ($\eta_м$), РС и $МОП_1$ ($\eta_р$), в УТ ($\eta_т$) и эффективность модуляции опрашиваемого светового пучка регистрирующей среды $\eta_с$. В реальных оптических системах по оптимистическим оценкам коэффициент η_2 не превышает 0,1.

Таким образом, величина $t_э$ ограничивается с одной стороны реальными возможностями мощности матриц лазеров в соответствии с соотношением (5), а с другой стороны допустимым теплоотводом матрицы фотоприемников (соотношение (6)). При $P_т$ = 5 Вт/см^2, $\rho = 10^6$ бит/см^2 и $W_1 = 10^{-14}$ Дж матрица фотоприемников не позволяет вести опрос чаще, чем через одну наносекунду.

При световой мощности одного лазера в матрице ПМЛ Рсл = 10^{-2} Вт, η_2=0,1, n = 50 ограничение со стороны лазеров имеет ту же величину. В этом случае электрическая мощность, потребляемая матрицей лазеров (устройством), с учетом потерь в лазере η_1, определяется как

$$P = Pсл\, Q / \eta_1 \quad (7)$$

и составляет 10^4 Вт.

Сокращение объема памяти до $Q = 10^4$ и увеличение времени до 10 нс приведет к потреблению 0,1 кВт электрической мощности на модуль АПК. В этом случае t_o уменьшится до 0,5 нс, а $P_{СЛ} = 10^{-3}$ Вт.

Теперь определим возможное время записи в АПК - $t_з$. В режиме записи информации возможны две ситуации: одна из них - адрес ячеек памяти, по которому должны быть записаны признак и относящаяся к нему информация, известен, другая - адрес свободных ячеек не известен. Понятно, что во втором случае время записи $t_з$ увеличится на величину времени выявления адреса свободных ячеек памяти, несколько большую, чем $t_э$.

Время цикла записи $t_э$ при известном адресе ячеек определяется быстродействием используемой в АПК адресной системы оптической выборки ($t_{ОВ}$), временем записи информации в УТ ($t_{УТ}$) и инерционностью переходных процессов в РС ($t_{рс}$). Как правило, $t_{ОВ} + t_{УТ} < t_{рс}$, поэтому $t_э \approx t_{РС}$.

При использовании в качестве регистрационной среды реверсивных светочувствительных материалов или сэндвичевых структур на основе жидких кристаллов с пороговым изменением оптических свойств, характеризующихся

энергетической чувствительностью $E_п$, время записи парафазного словах числом разрядов n может быть оценено с помощью следующего соотношения:

$$t_э = 2n\, E_п / \rho \eta_м\, \eta_т\, P_{СЛ} \qquad (8)$$

Полагая, что в настоящее время вполне реальным можно считать $E_п=10^{-5}$ Дж/см^2 , а величину потерь в оптической системе ($\eta_м$, $\eta_т$) не превышающей 0,5, определим $t_э$ для интересующих нас параметров $P_{сл}= 10^{-9}$ Вт, $\rho = 10^6$ бит/см^2 и n = 50. Время записи в этом случае не будет превосходить 2 мкс.

Таким образом, создание светочувствительной РС с временами записи и стирания, находящимися в субмикросекундном диапазоне, является предметом поиска.

Однако, проблемы записи и стирания информации в сэндвичевых структурах с жидкокристаллическим слоем упрощаются в случае обеспечения режима групповой записи по 100 и более слов (с временем записи $t_{з.гр}$). В этом случае стирание информации может происходить на фоне работы памяти за счет автономной электрической системы. Энергетика записи может быть существенно снижена за счет подвода электрической энергии извне; при этом за счет только групповой записи, например по $Q_{гр}$ = 100 чисел, среднее время записи $t_{з.ср} = t_{з.гр} / Q_{гр}$ может быть сокращено до 20 нс.

Подводя итог проведенного анализа, можно сделать следующие выводы:

а) необходимый объем памяти при удовлетворительном темпе ассоциативного поиска может быть достигнут только путем объединения нескольких модулей ассоциативной памяти в единую систему;

б) требуемый средний темп записи $t_{з.ср}$ может быть реализован только при групповой постраничной записи информации, а для достижения времени записи $t_З$, равного $t_п$, должны быть найдены специальные схемотехнические решения;

в) целесообразно модуль ассоциативной памяти разбить на два блока: ассоциативного поиска ключей (АПК) и хранения соответствующих этим ключам данных (ПД);

г) блок ассоциативного поиска целесообразно реализовать с использованием оптических принципов, а ПД на полупроводниковых интегральных схемах.

2. Возможные блок схемы построения ассоциативной памяти высокого быстродействия и большого объема

В настоящем разделе делается попытка построения АП с заданными параметрами с использованием имеющихся научно - технических достижений в оптике и электронике. Основные параметры АП: $Q_{апк} > 10^6$ ключей, $Q_{зуд} > 10^6$ слов, $n_{апк} = 10^2$ (парафазные 50-разрядные ключи), $n_{зуд} = 10^2$, $t_п = t_З = t_{З.ср} = 1$ нс ($П \geq 10^9$ оп/с).

Чтобы обеспечить заданные параметры, ассоциативная память должна состоять из 100 модулей, отвечающих поставленным требованиям. При этом необходимо решить две независимые друг от друга задачи. Поскольку возможные параметры одного модуля такой памяти по состоянию на сегодняшний день, как следует из раздела 1, могут быть следующими: $Q_{апк} = Q_{зуд} = 10^4$,

$t_п$ = 10 нс, $t_э$ > 2 мкс, то для создания АП с заданными параметрами требуется обеспечить эквивалентное время записи информации в память ключей, не превышающее времени $t_п$. Второй задачей является организация такого алгоритма работы модулей памяти, при котором переполнение одного или нескольких модулей не приводило бы к остановке корректного функционирования всей АП, а прекращение работы происходило бы только при переполнении всего объема памяти, равного Q = 10^6 слов.

2.1 Возможные решения первой задачи

Попробуем решить первую задачу внутри каждого модуля путем использования буферной ассоциативной памяти,, построенной на базе полупроводниковой схемотехники и обладающей на порядок меньшей емкостью, чем АПК, но требуемыми временами записи $t_З$ и поиска $t_п$. Введение такого буфера позволяет накопленные в нем ключи в случае необходимости переписывать групповым способом в основную оптическую ассоциативную память ключей (ОАПК). Групповая запись может включать по 100 и более ключей, что дает возможность среднее время записи ключа $t_{З.СР}$ сделать равным времени поиска $t_п$. Блок-схема такого модуля изображена на Рис.5. При реализации этой схемы предполагается использование промышленно выпускаемых интегральных схем.

В качестве ассоциативной полупроводниковой памяти (ПАПК) используем регистровую память со временем поиска по ключам 10 нс и суммарным объемом 500 1000 слов. Ключ и данные в ПАПК находятся в одном и том же регистре. Объем памяти ПАПК во многом зависит от времени записи страницы оптической ассоциативной памяти ключей ОАПК.

Обращение в ПАПК осуществляется посредством опроса ключей всех регистров памяти. В случае совпадения одного из ключей выдается соответствующее ему данное из того же регистра. Кроме того, запись и считывание в ПАПК могут происходить по указателям $УК_1$, $УК_2$ и $УК_3$. Считывание данного сопровождается стиранием всей информации регистра, что осуществляется установкой "I" в поле меток. По указателю $УК_1$ происходит запись искомого ключа и данного из входного регистра на свободное место, после чего указатель передвигается на следующее свободное место. По указателю УК происходит считывание регистров для формирования буфера записи (БЗ). Считывание осуществляется последовательным опросом поля меток регистров. При наличии в регистрах данных (метка в "0") данные считываются и последовательно заполняют регистры буфера записи. В том случае, если данное отсутствует (метка в "1"), указатель перемещается на следующий регистр, а метка устанавливается в "0". По заполнении буфера записи (100 слов) считывание прекращается и начинается групповая запись информации БЗ в памяти ключей ОАПК и данных в память данных через управление групповой записью. Запись производится на свободное место по адресу, получаемому из буфера свободных адресов.

После записи информации из буфера в поле ключей ОАПК происходит открытие этого поля для поиска и соответствующее уменьшение поля поиска в ПАПК с зоны между $УК_2$ и $УК_3$ до зоны между $УК_1$ и $УК_2$.

По $УК_3$ происходит считывание ключей тех регистров, обращение к которым по поиску ключей произошло после их записи в БЗ - только они в пространстве между $УК_2$ и $УК_3$ имеют в поле меток "I". Эти ключи передаются в буфер ключей для стирания ненужных данных в ОАПК и в память данных. Считывание ключей по $УК_3$ происходит с момента окончания записи информации из БЗ в ОАПК и память данных и заканчивается после того, как значение $УК_3$ не будет равно значению $УК_2$. После окончания этой операции может начаться считывание данных в БЗ по $УК_2$ и так далее. Заполнение БЗ может быть приостановлено в том случае, если значение $УК_2$ сравняется со значением УК), и продолжится, как только значение $УК_1$ увеличится.

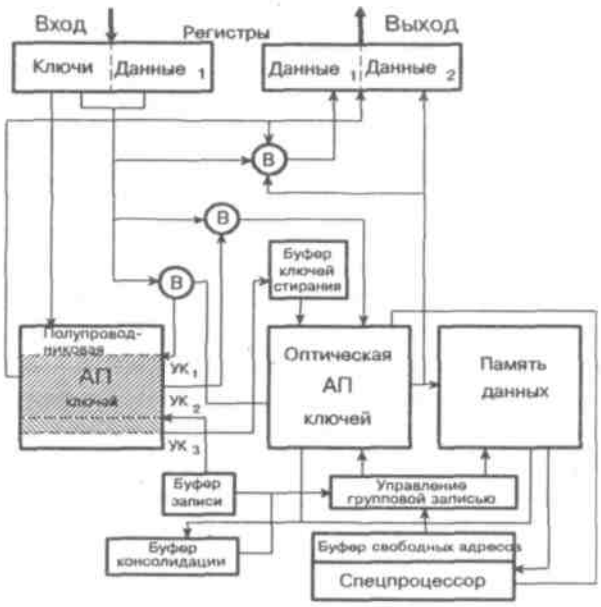


Рис.5. Модуль ассоциативной памяти.

Рассмотрим последовательность работы такого модуля АП. Ключ и данное для поиска по ключу поступают на входной регистр, после чего ключ передается в ПАПК для ассоциативного поиска в нем необходимого данного. В том случае, если ключ найден, данное $Д_1$ через вентиль $В_1$ и данное $Д_2$ из ПАПК подаются на входной регистр и операция заканчивается.

В том случае, если ключ в ПАПК отсутствует, код ключа из входного регистра через вентиль $В_2$ передается в ОАПК для поиска в нем необходимого данного, и если ключ найден, то данное $Д_2$ из памяти данных поступает на выходной регистр, а данное $Д_1$ через вентиль $В_1$ выдается на выходной регистр. В том случае, если ключ в ОАПК не найден, данные входного регистра (К и $Д_1$) через вентиль ВЗ по указателю $УК_1$ записываются в ПАПК. Таким образом, модуль АП работает как обычная ассоциативная память с эквивалентным временем записи и считывания, не превышающим 10 нс **(**$t_з \approx t_п = 10$ нс**).**

Приостановка работы модуля может произойти только в том случае, если указатель $УК_1$ догонит указатель $УК_3$. Как правило, этого не происходит, если выполняется условие:

$$Rt_{з.ср} \leq t_з \ , \qquad (9)$$

где R - коэффициент разрежения записи в ОАПК, понимаемый как

$$R = П_{папк} / (П_{папк} + П_{оапк}), \qquad (10)$$

где $П_{папк}$ и $П_{оапк}$ среднее количество считываний из ПАПК и ОАПК при ассоциативном поиске данных.

Разрежение записей и соответствующее снижение требований к основным параметрам АПК: $t_з$, $t_п$ и Q, происходит, например, за счет того, что некоторые данные находятся в ПАПК и нет необходимости их записывать в ОАПК.

Естественно, что чем больше объем памяти ПАПК при одном и том же объеме памяти ОАПК, тем коэффициент R будет меньше и соотношение (9) в части $t_{З.СР}$ выполнить будет легче. По статистическим данным многих задач, выполняемых на ЭВМ, работающих по фон неймановскому принципу, АП объемом в 1 К слов почти в сто раз сокращает количество обращений к основной оперативной памяти.

В суперЭВМ новой архитектуры, число записей, если не применять каких-либо оптимизаций, должно быть равно числу считываний, поэтому коэффициент R становится значительно меньшим. Однако, можно надеяться на то, что ПАПК сократит обращение к основной памяти в несколько раз вполне возможно.

Возникает, безусловно, вопрос формирования адресов записи на свободное место. Учитывая, что скорость выдачи свободных адресов более чем на два порядка ниже скорости работы модуля АП, функции формирования свободных адресов постраничной записи могут с успехом выполняться спецпроцессором, который, получая физические адреса записи и считывания в ЗУД, должен следить за ее состоянием.

Алгоритм работы такого спецпроцессора может быть следующим. По мере освобождения страницы он выдает адрес первого слова страницы в буфер свободных адресов УГЗ. Ввиду того, что считывание данных из АПК и памяти данных происходит пословно, возможно такое состояние, когда свободных мест в памяти много, в то время как нет ни одной свободной страницы. В этом случае спецпроцессор производит консолидацию свободных мест в памяти, определяет страницы с наименьшим количеством данных и производит считывание их в блок консолидации (БК). После заполнения БК, объем которого может быть равен одной странице, спецпроцессор выдает команду на запись этой страницы из БК в АПК и память данных.

Работы по записи данных из БЗ и БК, а также стирание ненужной информации по ключам, указанным в буфере стирания (БС), могут выполняться на фоне основной работы модуля памяти. Для того, чтобы модуль АП работал с эквивалентной производительностью $П = 1/t_п$, необходимо выдержать соотношение (9) и условие $t_п = t_з = t_{зп.ср}$. В зависимости от времени записи в ОАПК несомненно будет сокращаться электронная составляющая аппаратуры модуля АП. В настоящее время ведутся интенсивные работы в направлении сокращения времени $t_{З.СР}$.

2.2 Возможные решения второй задачи

Рассмотрим теперь возможные способы решения второй задачи - объединения модулей в единую АП. Она распадается на две самостоятельные части. Во-первых, неоходимо обеспечить коммутацию приходящих данных по модулям АП так, чтобы не потерять возможности достижения высокой пропускной способности АП в целом, поскольку, например, сто модулей с темпом работы 10 нс каждый обладают пропускной способностью в 10 миллиардов слов в секунду (слово за 0,1 нс). Во-вторых, необходимо добиться такого распределения данных по модулю в течение решения всей задачи, чтобы вероятность переполнения любого из модулей была минимальной, а в случае возникновения такой ситуации она не привела бы к прекращению работы всей системы.

Задача обеспечения высокопроизводительной коммутации данных со ста входных направлений к ста модулям АП в настоящее время может быть решена только с использованием оптических средств передачи и коммутации информации. Можно с достаточной уверенностью утверждать, что при решении задачи высокопроизводительной коммутации и передачи информации оптические принципы уже в настоящее время могут быть вне конкуренции по отношению к полупроводниковым средствам. Вопрос построения подобных коммутаторов требует самостоятельного рассмотрения и выходит за рамки настоящей статьи.

Задача равномерного распределения информации по модулям памяти не является новой для вычислительной техники. Методы распределения информации по модулям памяти с помощью так называемого метода "хеширования" могут быть полностью использованы и для обеспечения равномерного распределения информации по модулям АП. Отличие заключается в том, что неудачно выбранная функция "хеширования" разрядов адреса, определяющая модуль памяти, в старой архитектуре приводила к снижению производительности ЭВМ на некоторых задачах, в то время как в новой системе переполнение модуля АП вызывает прекращение работы комплекса в целом. Поэтому задача в полной мере будет решена в том случае, если функционирование АП прекращается только при переполнении всех модулей АП, в то время как переполнение отдельных модулей приводит к замедлению работы системы.

На Рис.6 приведена структурная схема АП, позволяющая реализовать работу АП без ее остановки в случае переполнения отдельных модулей МАП.

Принцип работы этой системы состоит в том, что в случае переполнения какого либо модуля, он производит запись в другой, заранее определенный модуль. Все модули имеют регистр переполнения, в котором каждый модуль отображается одним разрядом. В случае переполнения модуля и записи данных в другой модуль устанавливается "1" в разряд, соответствующий модулю, в который направляются данные для записи.

Поиск по ключу, адресованному в блок, из которого была произведена запись в другой модуль, происходит корректно, если алгоритм поиска будет следующим: выполняется поиск данного в модуле АП, к которому адресован ключ и данное. В том случае, если искомый ключ найден, пара данных выдается на выход и операция заканчивается.

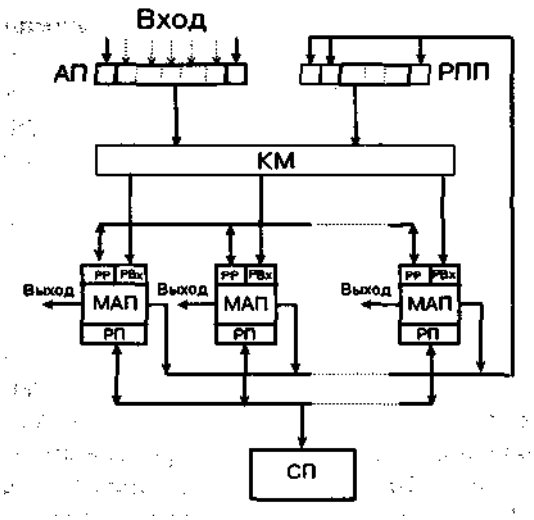


Рис.7. Модульная ассоциативная память.

АП - входной регистр; РПП - регистр повторного поиска; КМ - коммутатор модулей; МАП - модуль ассоциативной памяти; Вх - входной регистр модуля; РР - регистр результата повторного поиска; РП - регистр переполнения; СП - спецпроцессор.

В случае, если ключ в этом модуле АП не найден и в разрядах переполнения имеются единицы, ключ и данное отсылаются в те модули, в разрядах регистров переполнения которых находятся единицы. Данные для повторного поиска посылаются на регистр повторного поиска (РПП), после чего происходит обычная работа модуля АП. Если повторный поиск во всех модулях прошел, и поиск завершился успешно, то из одного модуля на выход выдаются необходимые данные и операция заканчивается. Если при первом поиске и нулевом регистре переполнения или при повторном поиске результат отрицательный, то происходит запись данных свободного регистра в тот модуль АП, куда они были первоначально адресованы.

Остается открытым вопрос, каким образом происходит стирание информациии в регистрах переполнения, если вся информация, которая была перезаписана из него в какой-либо модуль, в последнем при считывании уничтожена, и в повторном обращении к этой памяти нет необходимости? Так как нахождение в положении единицы какого-либо разряда в регистре переполнения в течение более продолжительного времени, чем это требуется, не приводит к ошибкам в работе АП, а лишь увеличивает иногда время поиска, процесс стирания единиц может выполняться спецпроцессором (СП) на фоне основной работы модулей. Для этого перезаписанная в модули ПАПК и ОАПК информация должна помечаться специальной битовой меткой, а в памяти данных каждое данное должно сопровождаться информацией, указывающей модуль, из которого это данное перезаписано. Спецпроцессор АП поочередно опрашивает в свободное от основной работы время модули АП и при необходимости осуществляет возвращение перезаписанных данных в свой модуль. Этот же СП, обмениваясь информацией со спецпроцессорами модулей АП, может

устанавливать в каждом модуле код адреса того модуля, в который необходимо передать информацию в случае его переполнения.

Приведенные на Рис.5 и Рис.6 схемы не претендуют на оптимальность решения поставленных задач. Авторы пытались найти компромисс между наиболее лаконичным изложением основных принципов построения подобных структур, их оптимизацией и детализацией.

Очевидно, что, применив принцип интерливинга, в блок-схеме Рис.5, можно запараллелить во времени работу поиска ключа в ПАПК и ОАПК, а в блок-схеме Рис.6 - работу по поиску информации в модуле, к которому адресован ключ, и процесс поиска того же ключа в модулях, указанных в регистре переполнения основного модуля. Может быть, более оптимально выбран компромисс между объемом модуля и его производительностью, если считать возможным запас по объему до 10^5 слов. Безусловно, должен быть оптимизирован критерий прекращения работы АП по переполнению (например, по переполнению группы из 8 - 10 модулей), что может существенно упростить структурную схему АП и так далее. Важно, что на современном уровне науки и техники можно создать АП с весьма хорошими параметрами по быстродействию и объему запоминаемой информации. Можно предполагать, что объем модуля АП вместе с питанием и системой охлаждения не будет превышать нескольких сотен кубических сантиметров, а при переходе от объемной геометрической оптики к интегральной можно ожидать, что общий объем занимаемый АП, состоящей из сотни модулей, не превысит 0,2 - 0,3 $м^3$ и АП будет потреблять не более 1 кВт электрической мощности.

Развивая идею совместного использования электронных и оптических принципов обработки информации в одном устройстве уже на более совершенном, развивающемся в настоящее время технологическом уровне, можно говорить о перспективной АПК, основой которой будет двумерная матрица ячеек памяти ключей, состоящих из пространственно - временных модуляторов света и размещенных в непосредственной близости от них и электрически связанных с ними триггеров, хранящих информацию о кодах ключей. На Рис.7. приведена схема такого модуля ассоциативной памяти.

Перезапись ключей по адресу в плате памяти, шифрация и дешифрация физических адресов ячеек памяти, а также хранение связанной с ключами информации осуществляется электронным устройством, имеющим в своем составе полупроводниковое ЗУ. Функции маскируемого регистра выполняет матрица полупроводниковых лазеров (МПЛ), на которую с помощью электронного дешифратора отображается инверсный код предъявляемого аргумента поиска. Физические адреса совпадений фиксируются матрицей фотоприемников. Оптическая система состоит из объективов, размещенных на фокусном расстоянии, двойного растра линз, поляризационно-чувствительных кубиков и расщепителя световых пучков каждого из лазеров на М световых пучков. Она позволяет проецировать в плоскость платы памяти микроизображения МПЛ и в плоскость МФП картину распределения s - компоненты излучения, возникающей за каждой из линз растра при отражении световых пучков от платы памяти.

Использование в качестве УТ полупроводниковой матрицы позволяет в мо-дуле АП реализовать схему одновременного поиска информации по многим ключам [11], что существенно расширяет пропускную способность модуля и АП в целом.

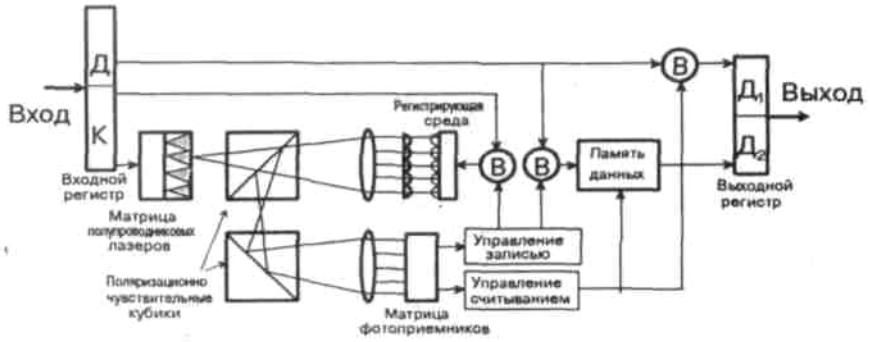


Рис.7. Оптоэлектронная ассоциативная память.

Практическая реализация в обозримом будущем оптоэлектронных АП с основными параметрами (емкость, быстродействие, темп обмена информацией), близкими к ранее приведенным, связана с решением целого ряда сложных технологических задач. Основная из них - разработка технологии, которая позволила бы разместить на единой плате памяти площадью порядка 10 $см^2$ интегральные электронные схемы управления и хранения информации и около 10^6 дискретных светомодуляторов. Нетривиален расщепитель световых пучков МИС, который для обеспечения одинаковой освещенности всех информационных ячеек должен при высокой оптической эффективности компенсировать неравномерность диаграммы излучения полупроводниковых лазеров и коррегировать аберрационные искажения (дисторсию) оптической системы и ряд других технологических трудностей.

Таким образом, можно предполагать, что в недалеком будущем оптические и электронные принципы обработки информации, реализуемые на единой технологической базе, будут дополнять друг друга.

Заключение

В настоящее время на базе современных технологических возможностей стало реальным создание ассоциативной памяти, отвечающей по быстродействию и объему хранимой информации требованиям перспективных архитектур суперЭВМ с производительностью 10^{10} - 10^{11} операций в секунду.

Дальнейшее развитие и использование оптических принципов обработки информации в архитектуре суперЭВМ нового поколения связано с существенной интеграцией оптических и электронных принципов на единой технологической базе.

Литература

1. В.С. Бурцев, Препринт ИТМ и ВТ АН СССР, М., 1977, N.32
2. В.С. Бурцев, Препринт Отдел вычислительной математики АН СССР, М., 1989, N.5.
3. J.R. Gurd et al., Comm. ACM, 1985, Vol.28, N.34.
4. Arvind, Culler D.E., In: Fifth Generation Computer Architecture, North Holland, 1985.
5. T. Shimada et al., Proceedings 13th Annual Symposium on Computer Architecture, ACM, 1986, P.226 234.
6. V.S. Burtsev, Optical Computing and Processing, 1992, Vol 2, N.3.
7. В.Б. Федоров, Всесоюзная конференция "Проблемы оптической памяти" Тезизы докладов и сообщений, М., ПИК ВИНИТИ, 1990, С.56 57.
8. В.Б. Федоров, В.Г. Митяков, Радиотехника, 1985, Vol.79, N.4.
9. В.К. Гусев, М.Л. Рослова, В.Б. Федоров, И.А. Шилов, 1982, Vol.22, N.6.
10. В.Б. Федоров, В.Г. Митяков, Оптика и спектроскопия, 1984, Vol 878, N.56.
11. V.S. Burtsev, V.B. Fydorov, Optical Computing and Processing, 1992, Vol.2, N.3, P.166.

Использование микропроцессоров традиционной архитектуры в системе потока данных

В.С. Бурцев, Л.Г. Тарасенко

Аннотация

Рассматривается возможность использования стандартных микропроцессоров или микропроцессорных наборов в качестве исполнительных устройств в новой нетрадиционной архитектуре суперЭВМ. Вводится понятие сложного оператора, выполняющего функции оператора системы потока данных. Обсуждается возможность обработки различных структур данных с помощью сложных операторов. На примерах показыается, что предлагаемые схемотехнические решения позволяют уменьшить требования к объему АП и снизить количество обращений к ней.

Введение

В статье рассматриваются возможности оптимизации схемотехнических решений суперЭВМ массового параллелизма с целью увеличения ее производительности за счет использования традиционных аппаратно-программных средств, не нарушая целостности основных концепций потока данных. Задача состоит в том, чтобы новый принцип организации высокопроизводительных вычислительных процессов не проигрывал бы по быстродействию традиционным принципам (фон-Неймана), даже на вычислительных конструкциях последовательного типа. Пределы производительности разрабатываемой нетрадиционной архитектуры, как это показано в [1], во многом определяются параметрами ассоциативной памяти: временем выполнения операции и темпом ее работы [4, 5]. Необходимо отметить, что создаваемая нетрадиционная архитектура

суперЭВМ позволит несколько ослабить требования ко времени выполнения операции АП по сравнению с традиционной архитектурой, построенной по принципу фон-Неймана. В то же время, требования к пропускной способности АП (темпу ее работы) остаются жесткими и определяют пределы быстродействия предложенной архитектуры массового параллелизма.

Пропускная способность одного модуля ассоциативной памяти ограничена разумными пределами потребляемой им мощности, исходя из возможности снятия тепла. При ограниченной предельной потребляемой мощности модуля объем его памяти будет обратно пропорционален максимальному темпу работы [5]. Дальнейшее увеличение темпа работы всей ассоциативной памяти требует новых схемотехнических решений, например, разбиения всей АП на достаточно мелкие модули. Однако, необходимо принимать во внимание тот факт, что при одном и том же объеме ассоциативной памяти уменьшение объема модуля повышает вероятность переполнения одного из них и усложняет коммутатор, а увеличение объема модуля повышает вероятность образования очереди заявок к модулю, что снижает общую производительность системы. Из этого следует, что чем больше снижаются требования к общему объему ассоциативной памяти, тем больший темп ее работы может быть реализован и тем больший предел производительности комплекса может быть достигнут.

Разгрузить ассоциативную память как по числу обращений к ней, так и по ее объему можно, используя следующие структурные решения в нетрадиционной архитектуре [1]:

- вместо векторов в АП хранятся только их дескрипторы;
- командная память вместе с константами вынесена из АП;
- ассоциативная память разбита на модули;
- имеется промежуточная буферная память БП готовых пар (токенов с одинаковыми ключами), сглаживающая темп работы АП и тем самым повышающая ее производительность;
- одновходовые операторы не проходят через АП.

В работе делается попытка еще дальше продвинуться в направлении сокращения требований к объему АП и, следовательно, увеличению темпа ее работы, используя статическую определенность и локальность данных некоторых вычислительных конструкций.

1. Сложные операторы

При представлении процесса вычислений в виде графа никак не определяется сложность оператора. В проекте, по умолчанию, под оператором понимается команда машины, имеющая один или два входных операнда и один или два выхода результата. Такое представление диктовалось тем, что аппаратно слишком сложно организовать ассоциативную память с тройным и более совпадением ключей операндов. Вторым ограничением было отсутствие памяти в скалярных исполнительных устройствах. Однако, концепция потока данных нисколько не изменится, если в качестве оператора будут выполняться, например, вычисления какой-либо из тригонометрических функций или любой другой функции. Если эта функция будет не более чем над двумя

параметрами, то для рассматриваемой структуры, кроме замены скалярного исполнительного устройства на микропроцессор с локальной прямоадресуемой памятью, по большому счету в аппаратной части менять ничего не надо Итак, условно вводятся два класса операторов: первый оператор - команда или простой оператор, второй оператор - функция или в дальнейшем сложный оператор. Аппаратно это не приведет к существенному увеличению объема оборудования, т.к. в принятой нами структуре каждое исполнительное устройство имеет все виды операций и командную память [1, 2].

Необходимо наложить определенные условия на тип выполняемых функций (сложных операторов). Так подпрограмма, реализующая сложные операторы, не может обращаться за результатом к внешней программе или сложному оператору, которые зависят от результатов или действий, выполняемых оператором.

В то же время, если оператор работает с несколькими входными операндами (более двух), то ограничений на их количество в принципе нет, так же как нет ограничений и на количество выходящих токенов результатов. Нет даже принципиального ограничения на ассинхронный запуск и выполнение сложного оператора. Выполнение сложных операторов может идти по мере поступления операндов (параметров) с выдачей результатов по мере формирования результатов.

Однако, для реализации ассинхронного запуска оператора с многими входными и выходными операндами необходимы определенные доработки в системе команд и принципах коммутации.

Пусть традиционный микропроцессор скалярного исполнительного устройства может при помощи подпрограммы, работающей в режиме фон-Неймана, выполнить как любой простой, так и сложный оператор. В этом случае микропроцессор должен иметь примитивную операционную систему для распределения оперативной памяти и достаточно развитую систему прерываний для организации вычислительного процесса по готовности программ к их выполнению.

Для реализации работы со сложными операторами (СО), кроме замены каждого скалярного ИУ на микропроцессор (МКП), необходимо ввести несколько специальных команд и изменить структурную схему суперЭВМ, введя еще один коммутатор типа $К_2$. На Рис.1 он обозначен $К_2$.

Коммутатор $К_2$ подключается параллельно коммутатору $К_1$ и может работать во времени либо последовательно, либо параллельно с последним в зависимости от входных параметров МКП. Параллельная работа потребует увеличения аппаратных затрат, но обеспечивает большую производительность всего комплекса.

Для запуска сложного оператора необходимо ввести специальную команду. Эта команда должна обеспечить инициализацию сложного оператора на одном из свободных микропроцессоров, включая выделение необходимой памяти, поставку входных параметров в нее и обеспечение выдачи результатов оператора. Очевидно, в одной команде все эти функции не всегда можно обеспечить, если иметь в виду поставку нескольких параметров и выдачу нескольких результатов в разные контексты выполняемой программы.

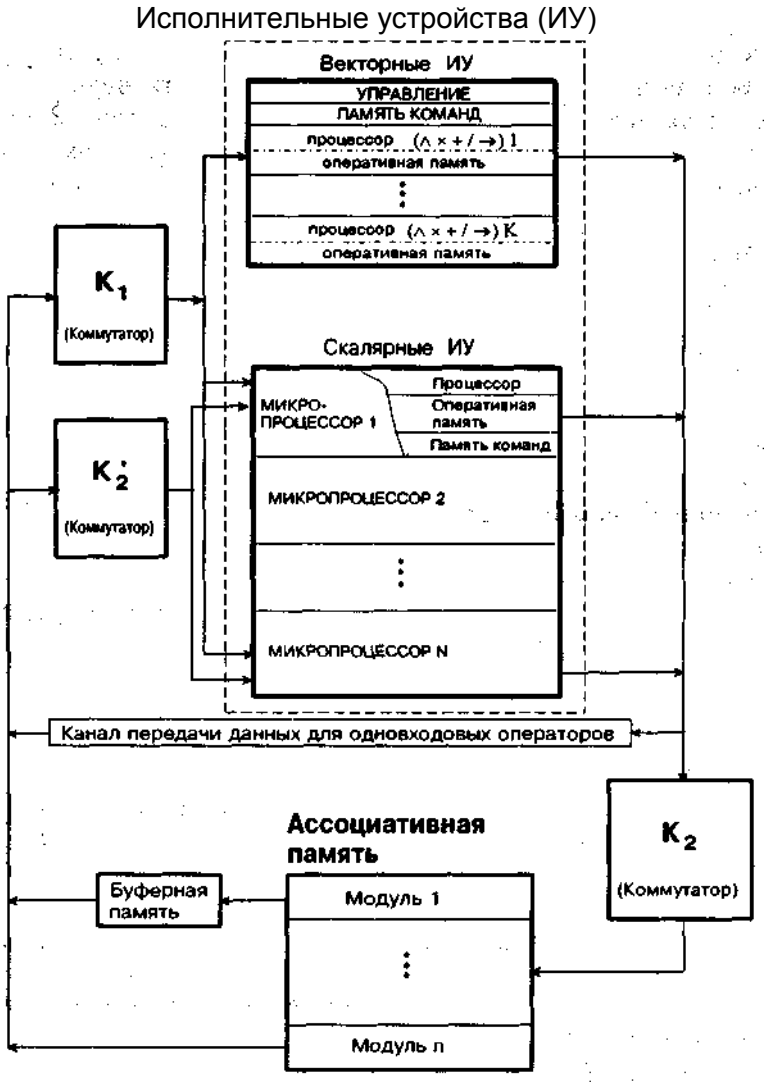


Рис.1. Структурная схема суперкомпьютера нового типа.

Однако команду, обеспечивающую начало этого процесса - "начало сложного оператора" (НСО), - реализовать можно. В то же время хотелось бы в простейшем случае при одном входном параметре и одном результате сложного оператора обойтись одной командой НСО (НСОl). Реализуем НСО в одновходовом варианте с целью экономии аппаратных средств. В этом случае результат любой команды может быть использован как первый операнд сложного оператора. Одновходовая команда, если есть ресурс процессора, выполняется непосредственно на нем, а если нет, подается на коммутатор $К_1$ для отыскания свободного процессора, готового к выполнению СО. Для того, чтобы осуществить

вход в подпрограмму, в качестве одного из адресов назначения в командах НСО и НСО1 должен быть адрес входа в подпрограмму сложного оператора, второй адрес может содержать адрес выдачи первого результата.

Задача поставки нескольких параметров в СО может быть решена, если правильно сформировать уникальное на данный момент динамическое "имя" СО. Использовать в качестве имени "цвет" или код контекста процедуры недостаточно, т.к. возможны обращения из разных мест этого же контекста к одной и той же подпрограмме. Поэтому ключ, состоящий из "цвета" и адреса входа в подпрограмму сложного оператора, в данном случае не годится. Для формирования ключа необходимо использовать адрес последнего оператора той программы, из которой осуществляется вход в подпрограмму сложного оператора. Для каждого входа в подпрограмму сложного оператора (СО) последней командой, выполняемой в той программе, из которой осуществляется вход, является команда НСО. Следовательно, уникальным именем сложного оператора для программы будет код ключа, состоящий из контекста этой программы (текущий контекст) и адреса команды НСО в этой программе. В простейшем случае, когда сложный оператор работает в контексте запускающей его программы и имеет один входной параметр и один результат, можно обойтись одной командой типа НСО для обращения к сложному оператору (НСОl). В качестве входного параметра в этом случае может быть использован результат предыдущей операции, а в качестве адреса результата код индекса или адрес назначения. Остальные поля "цвета" подпрограммы П и Т определяют контекст подпрограммы СО, в котором она работает.

Перечислим основные функции, которые должна выполнить команда НСО:

1. Определить МКП, в котором может быть выполнен сложный оператор, т.е. установить наличие блока оперативной памяти (64 или 128 слов) и, в первую очередь, того процессора, на котором выполняется формирование входного токена для НСО.
2. Поставить имеющийся параметр, сформировать и записать в память ключ для поставки первого результата по адресу назначения.
3. Сформировать и выдать на выход ИУ токен запроса параметров.
4. Запустить подпрограмму сложного оператора.

Возможен вариант команды НСО, содержащий в поле данных входного параметра суммарное количество параметров и результатов СО, которое заносится в токен запроса для последующего его стирания из ассоциативной памяти.

Токен запроса должен включать:

- в поле ключа: адреса назначения - адрес входа в команду НСО и "цвета" (текущего контекста);
- в поле данных: физические адреса номера микропроцессора, номер выделенного блока и номер слова;
- в поле операции: код специальной двухвходовой операции индексации;
- в поле управления памятью: признак, который предписывает АП вести поиск по ключу без учета поля индекса ключа и, возможно, число запросов.

При формировании хеш-функции токена запроса поле индекса не учитывается.

Токен запроса выдается в АП. Аппаратная часть работы самой команды НСО заканчивается записью токена запроса операндов в АП и входом в подпрограмму сложного оператора. В случае одного параметра и одного результата работа НСО значительно упрощается, так как нет необходимости в формировании токена запроса.

Необходимо отметить, что в том случае, если команда НСО не может быть выполнена на том же ИУ, где эта команда вызвана из-за отсутствия ресурсов, нужно сформировать пакет для выполнения этой команды на другом процессоре. В этом случае в качестве адреса назначения должен быть поставлен адрес самой команды НСО.

Поставка нескольких параметров производится командами подготовки индексации (ПИ), индексацией с записью (ИЗ) и индексацией со считыванием (ИС). В качестве параметра при выполнении этих команд в поле данных передается либо величина входного параметра (ВП), выполняемого СО, либо ключ (Кл), по которому СО должен поставить результат. В первом случае работает команда ИЗ, во втором - команда ИС. Значение параметра и ключ результата различаются признаком в поле индекса.

В том случае, если поставки параметров идут в том же контексте, в котором работает НСО, используется одновходовая команда подготовки индексации (ПИ). Если необходимо поставить параметр из любого контекста, то используется двухвходовая команда ПИ.

Одновходовая команда ПИ в качестве контекста так же, как и команда НСО, использует поля П и Т, а поле И определяет номер слова в блоке ОЗУ микропроцессора (величину индекса Nсл и признак ключа П). Признак ключа П, устанавливаемый в одном из разрядов поля индекса, определяет содержание данных ВП или Кл и соответствующее изменение кода операции в выходном токене, которую выполняет одновходовая команда ПИ. Ее действие практически сводится к изменению кода операции в формируемом токене. Код операции НСО заменяется на код операции "индексация с записью" (ИЗ), если в поле данных передается величина параметра (ВП), и на код операции "индексация со считыванием" (ИС), если в поле данных передается ключ, определяющий адрес и контекст передачи результата (Кл).

Двухвходовая команда ПИ из АП получает пакет, состоящий из контекста НСО ([ПТ]), индекса (И=Nсл&П) и параметра, содержащего либо величину параметра (ВП), либо ключ для передачи результата (Кл).

Команды ПИ в качестве адреса назначения имеют адрес команды НСО. Для формирования токена операнда, который должен найти в АП токен запроса, команда ПИ должна в поле ключа поставить ПТ текущего контекста. В этом случае поле ПТ и К сформированного токена будет определять имя (Р) сложного оператора в выполняемой программе. В поле данных формируемого токена операнда необходимо поставить параметр либо ВП, либо Кл. В токене операнде должен содержаться признак сохранения парного токена.

Операции ИЗ и ИС работают как обычные двухвходовые команды на любом свободном процессоре:

1. Физический адрес поля данных токена запроса индексируется: И=Nсл. Результат индексации устанавливается в поле [ИПТ] токена результата.

2. Адресом назначения K_1 является вход в подпрограмму.

3. В поле данных токена результата устанавливается величина параметра (ВП) в случае выполнения команды ИЗ или код ключа Кл в случае выполнения команды ИС, в поле кода операции устанавливается код команды продолжения сложного оператора ПСО.

4. Сформированный токен через K_2' передается на выделенный микропроцессор для выполнения команды ПСО, которая осуществляет передачу управления по адресу K_1 на подпрограмму, выполняющую запись параметра или запись со считыванием результата и передачу управления на продолжение СО, адрес которого содержится в первом слове блока ОЗУ.

Как уже упоминалось, микропроцессор имеет мультипрограммный режим с развитой системой прерываний. В том случае, если для выполнения подпрограммы нет необходимых данных, выполнение этой подпрограммы прерывается, адрес прерывания запоминается в первом слове блока ОЗУ и процессор может перейти к выполнению подпрограммы любого другого сложного оператора, готового к выполнению. Необходимо отметить, что все ранее описанные команды скалярного процессора, кроме групповых, не требуют прямоадресуемых регистров и счетчика команд для их реализации, что дает возможность упростить прерывание для беспрепятственного прохождения простых скалярных команд: достаточно приостанавливать выполнение программы сложного оператора без сброса в память регистров и счетчика команд.

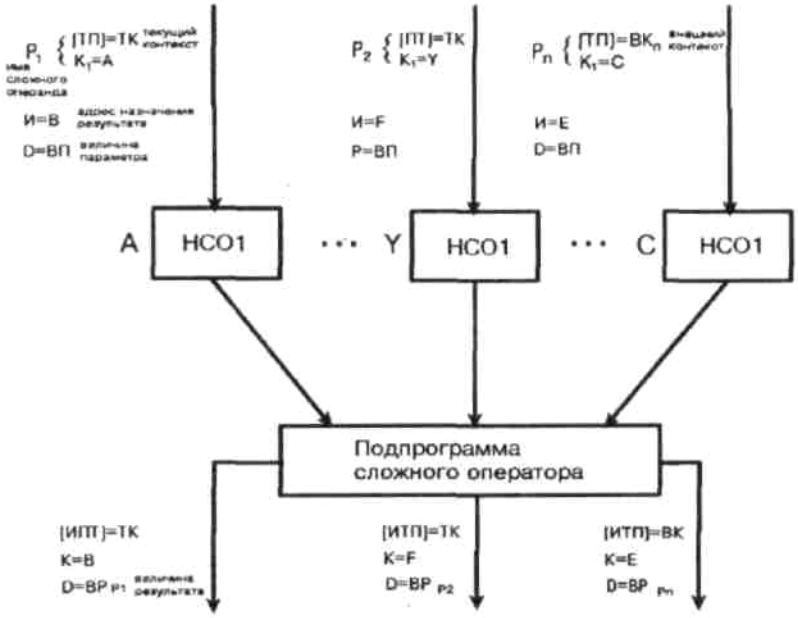


Рис.2. Работа со сложным оператором с одним входом и выходом.

Проследим последовательность работы сложного оператора в простейшем случае при одном параметре по входу и одном результате (НС01) (Рис.2).

Имя программы (Р) в любом контексте определяется полями П и Т контекста выполняемой программы и кодом адреса назначения оператора НС01, который в каждом месте входа в СО разный. Необходимо по возможности исполнить столько НС01, сколько входов в сложный оператор может быть осуществлено за определенный период решения задачи. Поле И контекста может использоваться для определения адреса назначения выдачи результата И=К=В.

Команда НС01 выполняет следующие функции:

1. Проверяет наличие свободного блока памяти на выполняемом микропроцессоре, и, если его нет, формирует пакет для выполнения этой команды на другом процессоре. Поиск процессора осуществляется через коммутатор K_1.
2. Осуществляет вход в подпрограмму сложного оператора, которая выполняет запись единственного параметра из поля Д и ввод ключа, по которому может быть поставлен результат (поля П и Т текущего контекста и И=К=В или второй адрес назначения команды).
3. Переходит к выполнению самой программы сложного оператора, которая может прерываться выполнением простых операторов.
4. По окончании работы подпрограммы сложного оператора формируется токен результата, и если следующий оператор двухвходовой, он передается в АП.

Работа сложного оператора с многими параметрами по входу и выходу несколько отличается от вышеописанной (Рис.3). Сложность состоит в том, что необходимо запомнить физический адрес микропроцессора и блока ОЗУ в нем, выделенных для выполнения сложного оператора. Работа этой команды, так же как и команды НСОl, начинается с определения свободного процессора для выполнения СО. Отличие работы команды НСО состоит в том, что для поставки нескольких параметров она должна сформировать токен запроса с тем же ключом, что и ключ входа в оператор НСО, т.е. с адресом назначения НСО. Только в этом случае можно организовать вход по всем параметрам, используя одно и то же имя (ключ) сложного оператора Р.

Токен запроса состоит из полей П и Т, текущего контекста и адреса команды назначения, в данном случае адреса А команды НСО (имя сложного оператора Р). В поле данных (Д) устанавливается физический номер процессора и блока ОЗУ (ФА), в котором СО выполняется. Сформированый токен запроса посылается в АП, т.к. команды назначения будут двухвходовыми (ИЗ и ИС). Остальные этапы работы НСО полностью совпадают с работой НСОl.

Подпрограмма СО может прерываться простыми операторами и может остановиться в ожидании остальных входных параметров. В этом случае процессор освобождается для выполнения других простых или сложных операторов. Все параметры, кроме первого, вводятся в подпрограмму СО через простые команды ПИ, ИЗ и ИС.

По приходу параметров, команды ПИ формируют токены для команд ИЗ и ИС, которые состоят из имени Р (поле П, Т и К|) и самого параметра в поле Д. Величина индекса берется из поля И контекста (И=Nсл&П), определяющего имя СО. Токен с операцией ИЗ формируется в том случае, если в поле Д

стоит ВП. Если в поле Д поставлен адрес и контекст назначения результата (Кл), формируется токен с операцией ИС.

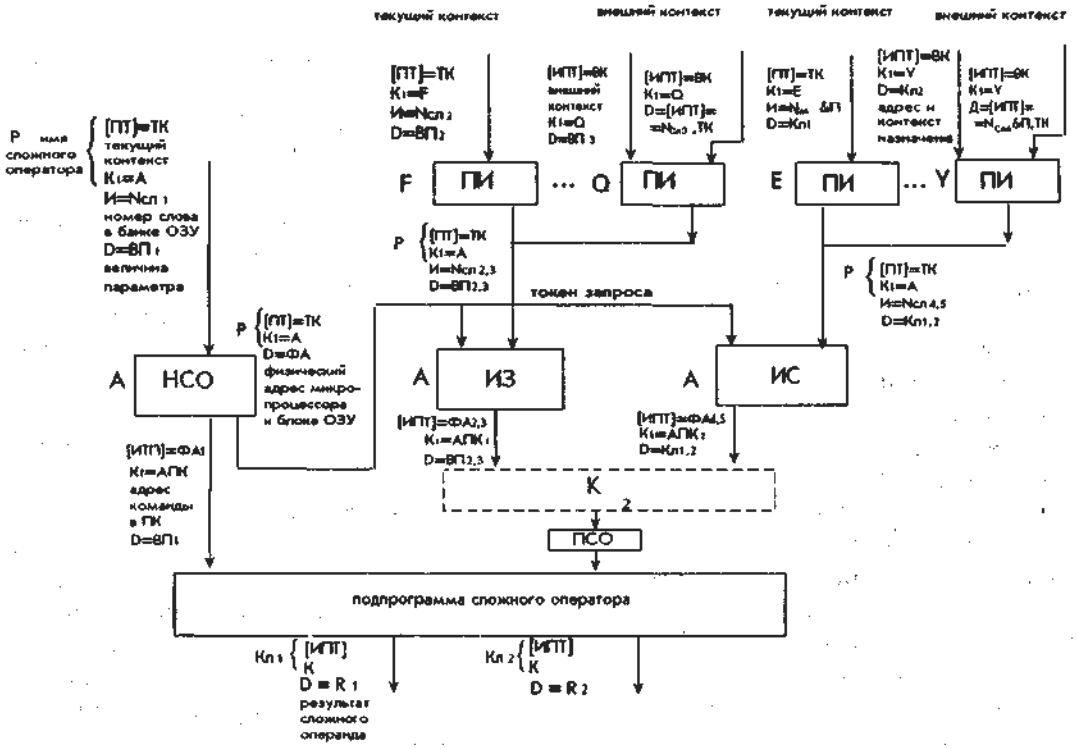


Рис.3. Работа со сложными операторами со многими входами и выходами

Двухвходовые команды ПИ используются тогда, когда необходимо ввести параметр из внешнего контекста по отношению к контексту НСО. В этом случае контекст СО и индекс И содержатся в поле данных первого операнда, а значение параметра в поле данных второго операнда. Сформированные токены операндов с операциями записи или считывания поступают в АП и по ключу Р объединяются с токеном запроса. Сформированные пакеты выполняются как простые операторы ИЗ и ИС. Эти операторы производят индексацию физического адреса содержащегося в поле токена запроса кодом индекса И и вместе со значением параметра посылаются через $К_2$' в тот процессор, на котором выполняется СО с именем Р. После того как подпрограмма СО получает результаты, она выдает их в соответствии с ключами $Кл_1$ и $Кл_2$, поставляемыми в качестве параметров посредством команды ИС.

Необходимо отметить, что введение сложных операторов возможно благодаря тому, что достаточно последовательно выдержан принцип обезличенного распределения всех видов ресурсов, таких как исполнительные устройства, коммутаторы $К_1$, $К_2$. Исключением являются только модули ассоциативной памяти. Этот принцип работы обеспечивается:

- алгоритмом работы коммутатора K_1, выдающего готовую к исполнению пару токенов на свободный процессор;
- исключением буферных регистров на входе исполнительных устройств скалярного процессора;
- системой равномерного распределения загрузки входных регистров K_1 с использованием буферов памяти готовых пар;
- алгоритмом ассинхронной работы коммутатора K_2;
- алгоритмом работы микропроцессоров исполнительных устройств, позволяющим исключить буферные регистры на их входах.

Любое отклонение от этого принципа при работе со сложными операторами может дать нежелательный простой оборудования. Так, если бы мы поставили буферные регистры на входе каждого исполнительного устройства, могла бы возникнуть ситуация, при которой в одном из исполнительных устройств достаточно долго выполняется сложная операция, а на его входе в очереди стоят готовые к исполнению пары токенов. В то же время имеется достаточное количество свободных микропроцессоров.

Введение сложных операторов может вызвать аналогичную ситуацию и в существующей структуре, но вероятность ее не так велика: Для того, чтобы сложный оператор вызвал задержку вычислительного процесса, аналогичную входным регистрам, необходимо два условия: первое - в микропроцессоре должна образоваться очередь готовых к выполнению сложных операторов; второе - среди стоящих в очереди готовых операторов должны быть такие, которые готовы выполнить вычисления сложного оператора до конца или вычислить хотя бы один промежуточный результат. Действительно, если в готовом к выполнению операторе нет данных для завершения оператора или получения промежуточного результата, то задержка его вычисления из-за очереди внутри одного процессора не скажется на вычислительном процессе. Таким образом, можно надеяться, что введение сложных операторов в проектируемую вычислительную систему не вызовет принципиальных задержек вычислительного процесса.

Введение сложных операторов позволяет:

- сократить требования к объему АП и количеству обращений к ней на определенных участках программ;
- упростить использование подпрограмм;
- увеличить производительность системы на последовательных вычислительных конструкциях с хорошо локализованными данными.

Первое положение не вызывает сомнения. Второе требует некоторого пояснения. Дело в том, что в системе команд, предложенной в [2], требуется приложить немало усилий со стороны системного программиста, чтобы из любого текущего контекста можно было войти в уникальный для подпрограммы контекст и по выполнению ее вернуться к прежнему текущему контексту. Задача сводилась к тому, чтобы присвоить оригинальный на момент выполнения подпрограммы "цвет" параметрам, а результат вернуть в необходимое место запускающей программы с восстановлением старого цвета. Эта задача не из легких, т.к. увеличение пула различных "цветов" ограничивает возможности программирования, и зачастую приходится вести учет освободившихся "цветов".

Новый механизм входа в подпрограмму "сложный оператор" не требует изменения контекста содержащей его программы, эта процедура заменена отысканием свободного места памяти микропроцессора, которое происходит аппаратно.

Третье положение (увеличение производительности) не является однозначным, т.к. можно привести примеры, когда производительность от введения сложного операнда будет падать. Покажем на примерах преимущества новой структуры машины для различных фрагментов программ.

Пусть необходимо вычислить:

$Y=X+X^2+X^3+X^4+X^5+X^6$;

Для исполнения вычислений выражения в исходной модели можно использовать программу, граф которой изображен на Рис. 4.

Фрагмент содержит 10 двухвходовых команд. Следовательно, при исполнении фрагмента потери динамического ресурса ассоциативной памяти составят 20 обращений. Максимальные потери статического ресурса ассоциативной памяти составят 3 слова.

Для исполнения фрагмента в модифицированной модели вычислений фрагмент может быть оформлен в виде сложного оператора с одним параметром и одним результатом. Для его запуска может быть использована команда НС01. Для исполнения фрагмента достаточно иметь 4 регистра, один из которых сумматор. Ни динамический, ни статический ресурс ассоциативной памяти использован не будет.

В следующем примере необходимо вычислить:

```
float x, y, z;

……..
for (I:=0;  i<N;  i++)
{x:=yz+x;  y:=xz+y;  z:=xy+z;}
```

Пусть N-константа.

В исходной модели, для исполнения вычислений, тело цикла будет оттранслировано в граф, изображенный на Рис.5.

Тело цикла содержит 6 двухвходовых команд, исполняемых последовательно. Исключая потери на организацию цикла, статически развернем цикл. Фрагмент содержит 6N последовательно исполняемых двухвходовых команд. Следовательно, при исполнении фрагмента в исходной модели вычислений потери динамического ресурса ассоциативной памяти составят 12N обращений за считыванием и записью. Потери статического ресурса ассоциативной памяти в процессе исполнения фрагмента составят 5 слов.

Для исполнения фрагмента в новой модели вычислений он будет оформлен в виде сложного оператора с тремя параметрами и тремя результатами. Для запуска сложного оператора будет использована команда НСО. При исполнении потребуется 8 обращений к ассоциативной памяти за параметрами и результатами. Потери статического ресурса ассоциативной памяти составят 1 слово (для токена запроса параметров). Для исполнения фрагмента в микропроцессоре достаточно иметь 4 регистра, один из которых сумматор.

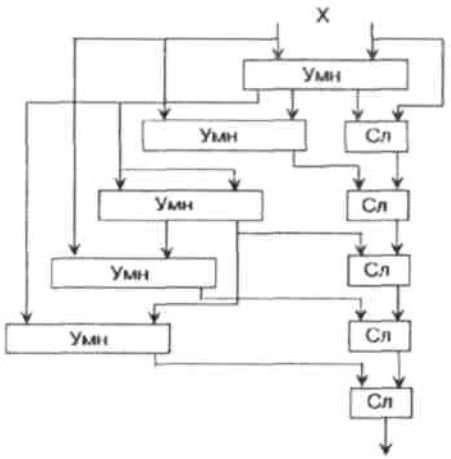


Рис.4. Граф программы

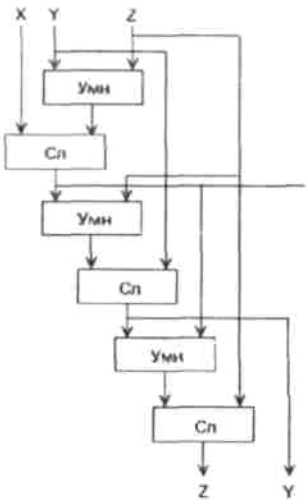


Рис.5. Граф программы

Итак, при использовании новой модели вычислений получим экономию динамического ресурса ассоциативной памяти, равную 12N-8 обращений, экономия статического ресурса составит по крайней мере 4 слова.

Приведенный на Рис.6 пример предполагает последовательный сбор параметров или элементов вектора в старой структуре. Этот пример интересен тем, что в новой структуре ввод параметров K_1 - K_n осуществляется по мере их готовности. Таким образом, новая структура имеет преимущество перед старой как по объему памяти, так и по времени выполнения этого фрагмента программы. Дело в том, что в случае задержки каких-либо параметров K; с малыми номерами, все параметры с более высокими номерами будут приняты в локальную память микропроцессора по мере их поступления и после прихода последнего без задержки будет сформирован вектор. Экономия по памяти составит в среднем n/2 - 1 ячейку АП. Этим методом можно воспользоваться для сокращения времени сбора вектора (см. Раздел 2).

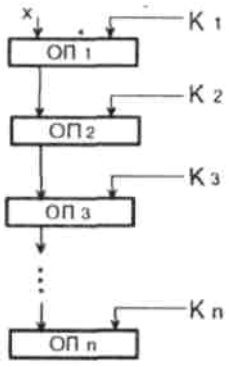


Рис.6. Граф программы

Из приведенных примеров видно, что в режиме потока данных неэффективно исполнять последовательные или "не очень" параллельные программы, так же как в режиме фон-Неймана неэффективно исполнять параллельные программы. Поскольку программы содержат как последовательные, так и параллельные фрагменты, то для их эффективного исполнения целесообразно создавать архитектуры, сочетающие оба принципа вычислений: как традиционный последовательный, так и параллельный, аналогичный принципу потока данных.

Ключевой проблемой при использовании сложных операторов в структуре потока данных является проблема разделения исходной программы на два класса: сложные и

простые операторы, поскольку при неудовлетворительном ее решении можно получить потери эффективности вычислений по сравнению с исходной структурой машины. Это может быть, например, в том случае, если граф, реалиующий оператор, имеет достаточную степень параллельности, и эта параллельность оказалась неиспользованной внутри сложного оператора.

Повышение эффективности вычислений в новой структуре может быть достигнуто в основном за счет сокращения числа обращений к ассоциативной памяти и увеличения коэффициента переиспользования данных внутри операторов.

Следовательно, для простых операторов, имеющих малый внутренний параллелизм и высокий коэффициент локализации по данным, имеет смысл выделять небольшой объем локальной памяти в МКП для выполнения СО.

2. Сборка и разборка вектора

Новый механизм выполнения сложных операторов может быть использован и для формирования вектора, передаваемого в векторное исполнительное устройство и для приема и выдачи элементов вектора в скалярную часть процессора. Эти две операции могут рассматриваться как сложные операторы. Для формирования вектора НСО находит процессор со свободной памятью и формирует токены запроса на все элементы вектора. Поступающие в АП элементы вектора объединяются с токенами запроса и по физическому адресу записываются в указанный процессор по соответствующему физическому адресу его оперативной памяти. По окончании сбора всех элементов вектор последовательным кодом выдается в векторное устройство. Аналогичным образом может быть произведено считывание вектора в микропроцессор, имеющий на данный момент свободную память, после чего последний производит выдачу необходимых элементов в ассоциативную память в требуемом контексте. Использование этого механизма выполнения сложных операторов освобождает от необходимости установки специальной регистровой памяти для целей сборки и разборки вектора [2].

3. Массивы

До сих пор шла речь о потоковых вычислениях со скалярными данными. Однако, в программировании мы сталкиваемся с необходимостью обработки составных данных - массивов, списков, записей и т.д. Рассмотрим, как можно идею потоковых вычислений распространить на обработку массивов, не нарушая ее концептуальной целосности.

Естественно, по дугам графа можно передавать целиком все элементы массива. Однако, можно заменить передачу массива передачей его описания или, вернее, ссылки на место его хранения. В этом случае при необходимости размножения массива осуществляется размножение ссылок на этот массив. Такие операции над массивами, как выборка элемента массива или изменение элемента массива, можно выполнять посредством индексации, т.е. указанием адреса на элемент массива при помощи индекса. В этом случае описание массива

должно содержать адрес начала массива (его имя) в выбранном адресном пространстве (виртуальном или физическом), шаг расположения элементов массива в этом адресном пространстве и количество элементов массива. Рассмотрим наиболее часто встречающиеся структуры массивов: векторный массив и различные модификации массива I структуры.

Векторный массив характерен тем, что все операции *над вектором идентичны* для каждого элемента этого массива. Это свойство используется в векторном исполнительном устройстве. Описатель вектора (дескриптор массива) в скалярном процессоре концептуально эквивалентен данному. Исключение состоит в том, что поскольку размножается не сам массив, а только ссылки на элементы, необходимо следить за количеством обращений к элементам массива для того, чтобы иметь возможность освободить занимаемую им память, в то время как обращение к скалярному данному предполагает, как правило, его стирание.

Массив I структуры характерен тем, что с каждым его элементом может проводиться любая операция, причем обращение к различным элементам одной и той же структуры может проходить из разных мест общей программы, работающих в разных контекстах. Предполагается, что запись элемента массива при его формировании может быть одноразовая. Всякая повторная запись в любой элемент массива является ошибкой программы и вызывает соответствующее прерывание. На один элемент массива I структуры, так же как и в случае вектора, возможно иметь несколько ссылок, т.е. многократное считывание. I структура может быть уничтожена в памяти после того, как все ссылки на все ее элементы реализованы. Как правило, работа с элементами массива начинается, не дожидаясь его полного заполнения. В этом случае, если к какому-либо элементу пришел один или несколько запросов по считыванию, а значения этого элемента еще нет, эти запросы должны быть запомнены до момента прихода значения. После поступления значения все отложенные запросы элемента должны быть обработаны.

Реализация работы с массивами подобного типа в системе потока данных затруднена тем, что в том или ином виде мы имеем дело не с графом, а с памятью - данные после их считывания не уничтожаются. Ранее проблема работы с массивами подобного вида была решена за счет использования дорогостоящей ассоциативной памяти. Попробуем реализовать работу с такими же массивами, используя принципы выделения оперативной памяти одного из микропроцессоров скалярного ИУ, как мы это делали в случае сложных операторов.

Отметим также, что на практике часто необходимо в определенные моменты решения задач заменять значения отдельных элементов и групп. В отличие от I структур мы решили разрешить выполнять такие операции при условии, что программист в этом случае полностью отвечает за правильность синхронизации вычислительного процесса по использованию данных.

Часто встречаются массивы, которые храняться в памяти от запуска задачи до ее окончания. Такие массивы I структуры будем называть статическими. Принцип взаимодействия со статическими структурами не отличается от работы с ранее описанными массивами. Можно сказать, что статические массивы

являются частным случаем динамических массивов рассматриваемой I структуры. Рассмотрим, в первую очередь, работу динамических структур размером в один блок памяти, который выделяется для ее выполнения одним из МКП.

Ввиду того, что динамические массивы могут возникать и уничтожаться в процессе решения задачи, работа с ними требует постраничной разбивки задачи. Это диктуется тем, что принятый нами максимальный массив в 10 тыс. слов без разбиения на страницы даст существенные потери в плотности его упаковки в оперативной памяти (потери на дискретность). Предположим, что динамические массивы разбиты на блоки по 100 слов в каждом. Рассмотрим случай, когда размер массива не превосходит одного блока. Работа с массивами в этом случае практически ничем не отличается от работы со сложными операторами с многими входами и выходами. Блок памяти инициализирует команды НСО в одном из процессоров, после чего формируется токен запроса и посылается в АП. Ключ входного операнда НСО без поля индекса является фактически динамическим именем этого массива. В поле данных операнда НСО может стоять величина, определяющая число обращений к этому массиву. Подпрограмма сложного оператора выполняет все необходимые действия по мере прихода параметров, которыми в этом случае являются либо значения элементов массива, либо запросы по считыванию этих элементов, т.е. выдачи результатов. Эта же программа может вести общий счет обращений к элементам массива и при необходимости осуществляет его стирание. Так, запись элемента массива ничем не отличается от записи параметра сложного оператора. Однако, в этом случае (в поле данных ВП) программным путем осуществляется выдача всех отложенных запросов этого элемента, если они были. При поступлении ключа выдачи результата выполняется считывание элемента массива. В этом случае при помощи подпрограммы осуществляется считывание элемента, если он имеется в памяти и выдача его по указанному адресу. Если элемента нет, то запрос запоминается в отдельном блоке памяти и выдается после прихода значения элемента.

Обращения к элементам массива по их записи или считыванию происходит так же, как и при работе со многими входами и выходами через команды ПИ, ПСО, ИЗ и ИС. В случае записи элемента используется команда ИЗ, в случае считывания команда ИС. В том случае, если обращение к элементу массива происходит из контекста, в котором он был инициализирован (контекст массива), используются одновходовые команды ПИ. Если происходит обращение к элементу массива из внешнего контекста, используется двухвходовая команда ИП. Работа команд ИП, ПСО, ИЗ и ИС ничем не отличается от их работы со сложным оператором. Элементы массива являются параметрами в подпрограмме сложного оператора работы с динамическими массивами (Рис.7).

Работа с массивами, размер которых более одного блока памяти, может быть выполнена программным путем. В этом случае программно в первую очередь выполняются все команды НСО всех блоков массива. Выполнение команд НСО может быть выполнено групповой командой. После записи в АП токенов запросов всех блоков массива работа с элементами массива ничем не отличается от описанной выше.

Работа со статическими массивами много проще, чем работа с динамическими I структурами. Так, например, статические массивы любой длины нет необходимости делить на блоки, а малые массивы увеличивать до размера блока динамических массивов. Размер массива может быть сокращен до одного слова, что дает возможность работать со всем объемом оперативной памяти процессоров как с прямоадресуемой.

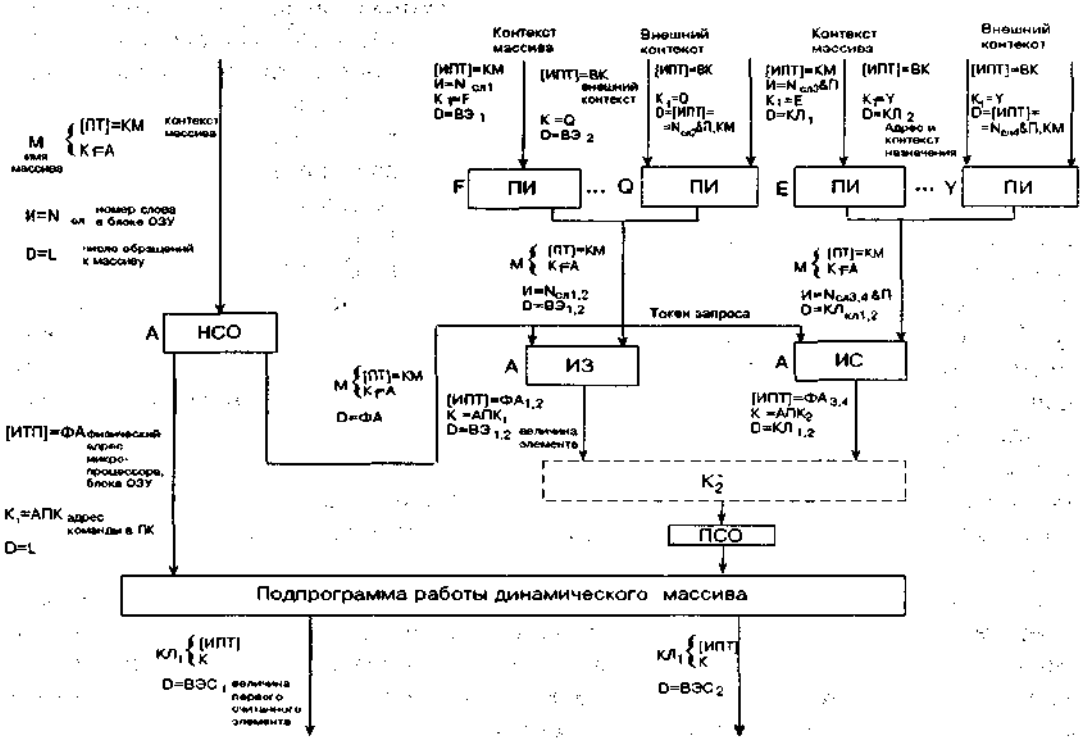


Рис.7. Работа с динамическими массивами

Работа со статическими массивами значительно упрощается за счет того, что имя массива и его структура известны при программировании и трансляции. По этому имени-адресу константы, хранящейся в ПК, из любого микропроцессора можно определить его физический адрес. Поэтому нет необходимости в Команде НСО, а следовательно и в команде ПИ. Обращение к элементам массива может осуществляться через команду индексации с записью элемента в статический массив ИЗст и команду индексации со считыванием элемента из статического массива ИСст (Рис.8).

Эти команды, так же как и команды ИЗ и ИС, производят индексацию физического адреса элемента величиной И=Nэл, указывающей на номер элемента в блоке. Отличие этих команд от команд ИЗ и ИС состоит в том, что они являются одновходовыми, т.к. физический адрес блока элемента они считывают из ПК по второму адресу назначения этой команды $К_2$. Первый адрес назначения указывает на вход в подпрограмму. Таким образом, имя подпрограммы

одназначно определяется адресом команды ИЗст и ИСст. Ввиду того, что эти команды одновходовые, их можно использовать из любого контекста программы. Таким образом, для обращения к элементу массива необходимо сформировать токен, в поле "цвета" которого определяется контекст, поле И определяет индекс элемента, а в поле данных стоит ВЭ в случае записи элемента и Кл в случае считывания элемента. Проиндексированный физический адрес элемента вместе с K_1 и ВЭ или Кл через коммутатор K_2' передаются на тот процессор, в котором хранится этот элемент массива. Дальнейшая работа определяется подпрограммой сложного оператора работы с массивами.

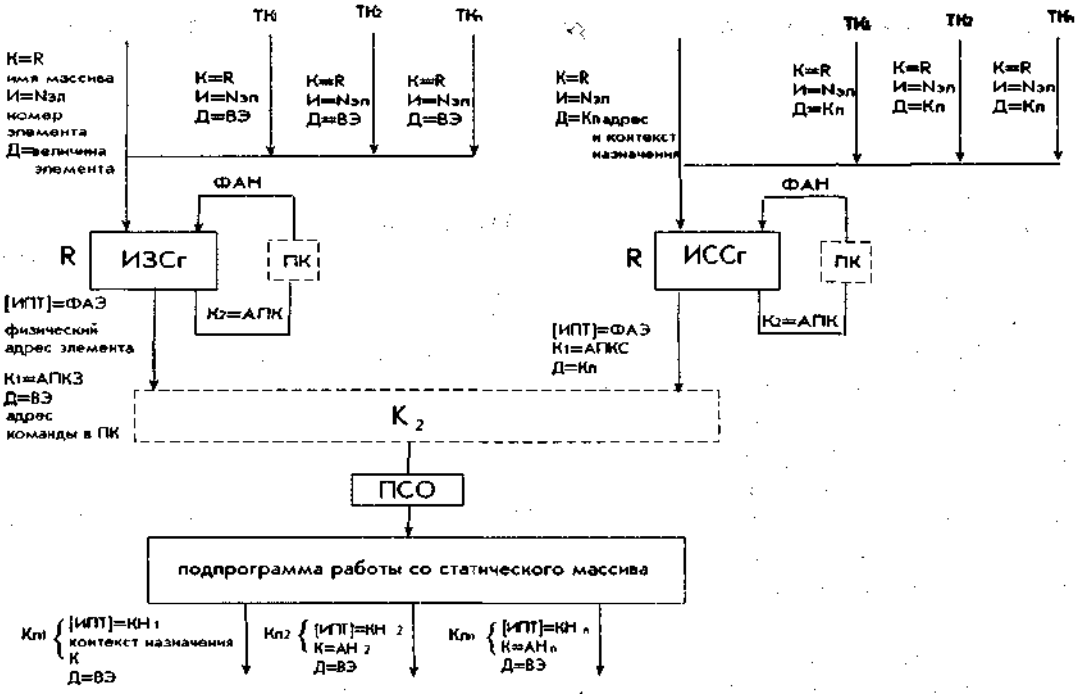


Рис.8. Работа со статическими массивами

Заключение

1. Приведенная архитектура может оказаться достаточно удачной для создания современных суперЭВМ массового параллелизма.

В предлагаемой нами системе массового параллелизма там, где вычислительный процесс последовательный с хорошо локализованными данными и программист может в статике эффективно запрограммировать этот участок вычислений, представляется возможность использовать сложные операторы, включая работу с массивами. Во всех остальных случаях распределение вычислительных ресурсов происходит в динамике по готовности параллельных процессов (пар токенов скалярных и векторных процессоров) и по готовности

к работе вычислительных ресурсов: микропроцессоров и их памяти, векторных процессоров и их локальной памяти, модулей АП и входов/выходов коммутаторов.

2. Описанная система массового параллелизма концептуально является однопроцессорной системой. Используя те же принципы потока данных, мы можем объединить эти процессоры в своеобразную многопроцессорную структуру (8-16 процессорную). Как и в традиционной многопроцессорной системе с целью достижения большей эффективности прохождения вычислительного процесса не будет желательно вмешательство человека для распределения процессоров и их АП по вычислительным процессам задачи.

Если сравнить предлагаемую архитектуру с традиционными, то можно увидеть ее интересное принципиальное отличие, состоящее в следующем. В традиционных системах АП в виде КЭШ памяти или безадресной памяти прямо-адресуемых регистров ставилась при процессоре и помогала активизировать вычислительный процесс на небольших локальных участках программы с малым количеством исполнительных устройств (например, так это реализовано в МВК "Эльбрус"). В новой архитектуре АП служит для глобальной оптимизации и синхронизации вычислительного процесса с использованием многих десятков исполнительных устройств, т.е. там, где человек не в силах справиться с этой задачей.

3. Предлагаемая система обладает целым рядом преимуществ конструкторско-технологического характера: она может быть построена на серийно выпускаемых микропроцессорах или микропроцессорных наборах, серийно выпускаемых высокоинтегрированных схемах ассоциативной памяти, интегральных схемах коммутаторов и оперативной памяти. Пропускная способность исполнительных устройств, коммутаторов и АП с целью достижения временного баланса этих устройств может регулироваться количеством микропроцессоров в каждом исполнительном устройстве, количеством параллельно работающих интегральных схем в модуле АП и количеством параллельно работающих коммутаторов $К_1$, $К_1'$ и $К_2$.

Дополнительным преимуществом даже однопроцессорной структуры является ее функциональная живучесть - выход из строя модулей ассоциативной памяти, микропроцессоров и коммутаторов не нарушает функционирования процессора.

4. Безусловно, разработка специальных исполнительных устройств для скалярных и векторных вычислений и использование оптических принципов широкополосной коммутации на порядок поднимет производительность однопроцессорного комплекса и позволит подойти к пределу производительности в 10^{12}-10^{13} оп/с, а на 8-16 процессорном комплексе достичь 10^{14} оп/с.

Авторы приносят благодарность профессору Д.Б. Подшивалову, сделавшему целый ряд принципиальных замечаний по публикуемой статье, многие из которых авторы учли в предлагаемой работе. Мы благодарны также И.К. Хайлову, Э.В. Сызько, Л.А. Козлову и М.Ю. Никитину, внесшим творческий вклад в эту работу.

Работа выполнена при поддержке Российского фонда Фундаментальных исследований.

Литература

1. В.С. Бурцев. Выбор новой системы организации выполнения высокопараллельных вычислительных процессов, примеры возможных архитектурных решений построения суперЭВМ (в данной книге).
2. Л.А. Богачева, Д.Б. Подшивалов. Проблемы построения системы команд для машин, основанных на принципе потока данных. В сб. "Вычислительные машины с нетрадиционной архитектурой. Супер ВМ." Вып.2, 1994, с.38-77.
3. Л.Г. Тарасенко. Реализация языков программирования на ЭВМ архитектуры потока данных. В сб. "Вычислительные машины с нетрадиционной архитектурой. Супер ВМ". Вып.2, 1994, с.108-125.
4. В.С. Бурцев. Тенденции развития высокопроизводительных систем и много-процессорные вычислительные комплексы. Препринт ИТМиВТ, 1977, с.27.
5. V.S. Burtsev, V.B. Fyodorov. Associative memory of new generation supercomputers based on optical information processing principles. Holography and Optical Information Processing, 1991, v.1731, p.201-216.

Особенности проектирования векторного исполнительного устройства в системе массового параллелизма с автоматическим распределением ресурсов

В.С.Бурцев

Аннотация

Исследуются особенности работы векторного исполнительного устройства (ИУ) в нетрадиционной структуре суперЭВМ. Описывается его базовая схема. Приводятся структурные схемы оптимизации векторных ИУ с описанием принципов их работы.

Введение

Введение векторного ИУ в состав новой структуры суперЭВМ обусловлено, в первую очередь, требованием уменьшения объема ассоциативной памяти (АП) скалярного процессора. Хранение векторных операторов требует большого объема памяти, в то время как необходимости в ассоциативной памяти для хранения этих данных нет. В ассоциативной памяти скалярного процессора достаточно хранить описание вектора (его дескриптор), имея в виду, что действия над всеми его элементами идентичны. В этом случае в поле данных токена скалярного процессора располагается дескриптор вектора. Объединенные в АП пары токенов дескрипторов посылаются в векторное устройство, где и происходит выполнение оператора над векторами, результат которого посылается в оперативную память векторного устройства. В качестве результата в скалярный процессор возвращается токен, у которого в поле данных устанавливается дескриптор вектора-результата. В этом случае полностью реализуется принцип управления потоком данных на векторных задачах. Некоторое отступление от этого принципа при работе с векторными операндами состоит

в том, что размножение вектора, которое требует пересылки всех его элементов, целесообразно заменить размножением его дескриптора. Это приводит к тому, что на один вектор "смотрит" несколько дескрипторов. Стирание в памяти элементов вектора после обращения к ним, как того требует принцип управления потоком данных, может вызвать искажение вычислительного процесса. Это обстоятельство вынуждает следить за количеством обращений к вектору в процессе работы, и только после того, как все "смотрящие" на него дескприпторы использованы, производить освобождение памяти. Соблюдая эти особенности обращения к векторам, с дескрипторами векторов можно работать как с данными [1]**.** Таким образом, роль скалярного процессора сводится к определению векторных операций, готовых к выполнению. Если средняя длина вектора равна 1000, то можно в тысячу раз уменьшить объем АП, работая с векторными операциями по такому принципу. Наряду с экономией дорогой, в аппаратном отношении, ассоциативной памяти мы получаем увеличение ее быстродействия, так как с увеличением объема производительность АП падает [2].

1. Постановка задачи

Предлагаемая новая архитектура суперЭВМ без векторного исполнительного устройства на скалярном процессоре реализует максимальный параллелизм вычислений при поэлементном выполнении векторных операций, так как в этом случае операторы программы работают по мере готовности каждого элемента вектора, что в векторном исполнительном устройстве практически исключено. Ограничением такого подхода реализации векторных операций являются аппаратные средства, которыми мы сегодня реально располагаем. В первую очередь это возможность реализации коммутирующих устройств и многопортовой ассоциативной памяти. Наиболее вероятными для реализации этих устройств являются оптоэлектронная и оптическая элементные базы. В настоящее время построены рабочие макеты коммутаторов на оптоэлектронной базе, превосходящие по параметрам полупроводниковые, и показаны возможности создания многовходовой ассоциативной памяти [3]. Возможно, что с внедрением этих устройств на оптических и оптоэлектронных принципах необходимость в векторных исполнительных устройствах отпадет.

Тем не менее в настоящее время на полупроводникой элементной базе может быть построена ассоциативная память объемом в несколько десятков миллионов ключей и созданы коммутаторы достаточно высокого быстродействия, осуществляющие широкополосную коммутацию 100х100 объектов за десятки наносекунд. Таким образом, можно считать, что на полупроводниковой базе может быть построена скалярная часть суперЭВМ, состоящая из 100 исполнительных устройств, работающих параллельно. Используя возможности скалярной части новой архитектуры по эффективной обработке сравнительно малых векторов (до 100 элементов), можно оставить за векторным исполнительным устройством работу с векторами, имеющими большое количество элементов (выше 100). Таким образом, достаточно просто решается проблема

падения производительности векторных исполнительных устройств при работе с малыми векторами, имеющая место в традиционных архитектурах.

В предлагаемой архитектуре максимальная длина вектора ограничена 10000 элементов. Ограничение вызвано только возможностями аппаратных средств.

Одновременно, структура векторного процессора должна быть по быстродействию хорошо сбалансирована со скалярной частью суперЭВМ. Важно правильно определить соотношение скалярных и векторных операций решаемых задач. Как показывает практика, число скалярных операций по отношению к операциям над элементами вектора в зависимости от решаемых задач составляет от 10% до 1%, то есть быстродействие векторного исполнительного устройства должно быть как минимум в 10 раз больше скалярного.

Исходя из этих соотношений, можно предположить, что объем векторной памяти в зависимости от задач должен быть в 10 - 100 раз больше скалярной.

Взаимодействие векторного ИУ со скалярной частью процессора по описанному принципу определяет некоторую особенность последовательности команд, поступающей на вход векторного ИУ, которая должна быть учтена при проектировании векторного устройства. Эта особенность состоит в том, что последовательность векторных команд, как правило, взаимонезависима. Другими словами, вероятность использования результата предыдущих команд в очереди значительно меньше, чем это имеет место в традиционных структурах. Это обстоятельство позволяет выполнять все команды одновременно или в конвейерном режиме. Напомним, что наиболее эффективно конвейерные процессоры типа Cray работают на взаимосвязанной последовательности векторных команд, так называемых "зацеплениях".

Мы рассмотрели один вид взаимодействия векторного ИУ со скалярной частью процессора нового типа, при котором управление векторным ИУ осуществляется посредством обмена токенами. Однако, структура суперЭВМ была бы неполной без возможности передачи вектора, сформированного в скалярной части, в векторную и наоборот. Естественно, этот обмен в зависимости от функций, выполняемых векторным ИУ, может сильно меняться по интенсивности. Дело в том, что при работе с векторами возникает необходимость выполнить операции между элементами самого вектора или осуществить трансфигурацию вектора. Все эти операции могут быть эффективно выполнены в скалярной части при условии достаточности ее ресурсов и обеспечении высокоскоростного обмена между векторной и скалярной частью суперЭВМ. В то же время выполнение межэлементных операций в самом векторном ИУ существенно снизит загрузку скалярной части и уменьшит обмен векторами между векторным ИУ и скалярной частью процессора.

2. Особенности структурной схемы векторного ИУ

Основой векторного ИУ является оперативная память векторов. От ее структуры во многом зависит архитектура всего векторного ИУ и его предельное быстродействие. Прежде всего, определим формат ОЗУ, то есть количество слов, считываемых за одно обращение к нему. Самое лучшее решение с точки зрения скорости работы, безусловно, сводится к формату ОЗУ, равному

самой большой длине вектора 10000 слов. Однако, это решение будет чрезвычайно неэффективно с точки зрения заполнения ОЗУ. Следовательно, вектор необходимо разбить на страницы, соответствующие формату ОЗУ. От выбора размера страницы существенно зависит коэффициент использования векторного ОЗУ ввиду потерь, связанных с малыми векторами и остатками от деления вектора на величину страниц.

Эффективность работы комплекса на векторах, меньших размера страниц, может быть обеспечена путем выполнения последних на скалярной части комплекса. Средняя величина вектора, безусловно, изменяется от задачи к задаче. Как показывает опыт, эта величина близка к 1000 элементам. В этом случае легко оценить потери ОЗУ на дискретность при различных длинах страниц. При размере страницы в 100 слов средняя величина потерь не превысит 5%, для 200 слов - 10%, для 400 слов - 20%. С учетом того, что операции над векторами размером меньше страницы будут выполняться на скалярной части комплекса, можно считать достаточной производительность векторной части в 10 раз больше скалярной. При этом темп работы векторной памяти должен быть в три раза больше темпа работы АП, так как каждая векторная операция требует двух считываний и одной записи. За счет увеличения формата векторной памяти до 400 слов, что потребует 400 модулей, мы получим увеличение производительности только в 4 раза. Увеличение производительности векторной части может идти и в направлении увеличения числа процессоров, работающих с каждым модулем памяти, с соответствующим увеличением числа блоков памяти в каждом модуле. Наиболее естественным представляется использование конвейерных процессоров с увеличением их количества до уровня конвейеризации в каждом процессоре и разбиением модуля памяти на блоки таким образом, чтобы элементы одного вектора располагались в разных блоках памяти.

Так, если число модулей N=400, то при 10 блоках памяти (M=10) можно для многих векторов получить увеличение производительности памяти, ее темпа работы в 40 раз по сравнению с темпом работы АП скалярного процессора, содержащего сто модулей. В этом случае коммутатор на выходе и входе модулей памяти и процессоров практически не требуется - он вырождается в сборку, а коммутация реализуется по временному принципу. Это наиболее реальный путь построения векторного ИУ, так как при одной и той же элементной базе темп работы скалярной части всегда будет соизмерим с векторной (Рис.1). Традиционный путь снижения числа обращений к ОЗУ за счет сверхоперативной памяти в данном случае малоэффективен ввиду того, что мы не имеем информации, даже вероятностной, на основании которой можно было бы принять решение о сохранении результатов в сверхоперативной памяти для дальнейшей работы: векторные команды, как было сказано ранее, выдаваемые скалярным процессором векторному для исполнения, никак алгоритмически не связаны между собой, так как их появление во времени непредсказуемо.

Таким образом, в векторном процессоре с каждым модулем памяти связан один или несколько векторных микропроцессоров. Число таких модулей равно формату ОЗУ и составляет 200-400 единиц. Управление процессорами происходит

от общего управляющего процессора, имеющего свою память команд ПК. Управляющий процессор принимает пакеты заданий от скалярной части, запускает векторный процессор и формирует дескриптор результата последнего аналогично тому, как это делает скалярный процессор.

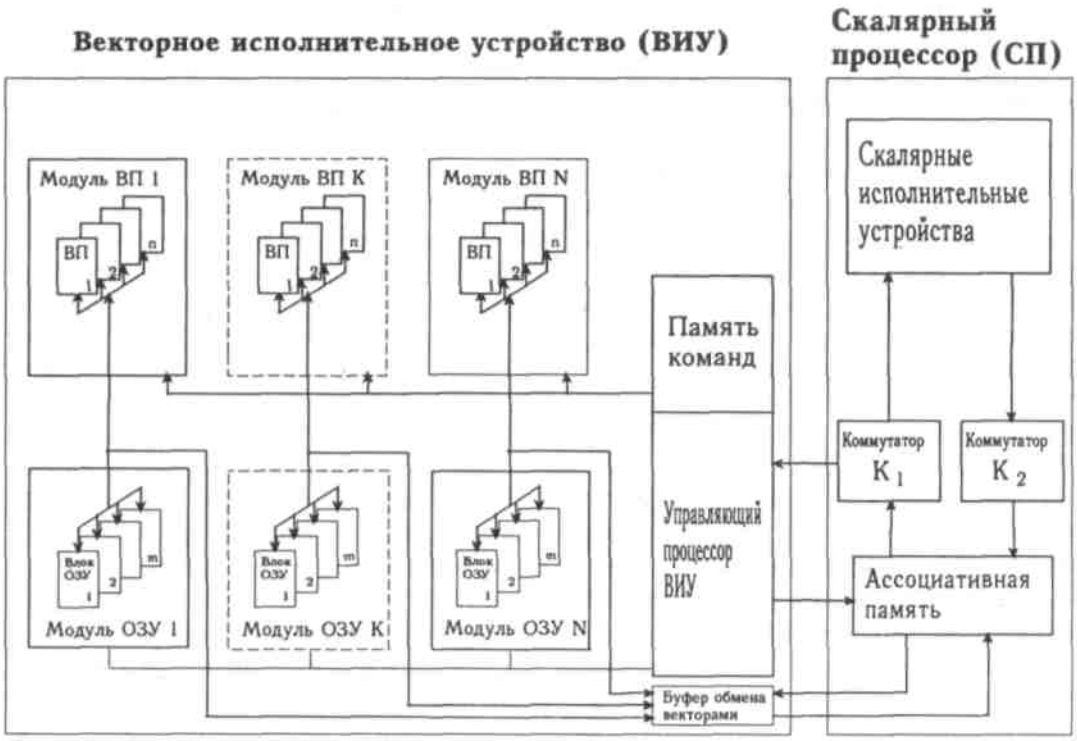


Рис.1. Базовая схема векторного процессора

Если вектор взаимодействует с постоянной величиной, управляющий процессор считывает ее из ПК и рассылает по всем векторным микропроцессорам.

Выполняя операцию обмена векторами между векторным ИУ и скалярным процессором, управляющий процессор считывает или записывает вектор в ОЗУ и через буфер векторов осуществляет обмен векторами между векторной и скалярной частями суперЭВМ. Управляющий процессор осуществляет контроль за количеством обращений к вектору и, если необходимо, удаляет его. Фактически управляющий процессор выполняет также функции операционной системы по распределению памяти (Рис.1).

Эта схема предусматривает выполнение всех действий между элементами внутри вектора на скалярной части суперЭВМ.

3. Оптимизация структуры векторного ИУ

3.1. Введение понятия вектора в векторное ИУ

Желательно сократить время выполнения операции над вектором, состоящим из нескольких страниц. Для этого надо сделать так, чтобы операции, начатые

над страницами двух векторов, не прерывались работой других векторов. В этом случае дескриптор должен описывать не страницу вектора, а вектор целиком, то есть его имя и количество страниц. В качестве имени может быть использован контекст. Управляющий процессор должен иметь справочник векторов, в котором каждому имени вектора соответствует список физических адресов его страниц. Опасение, что это замедлит вычислительный процесс за счет снижения его параллелизма, несправедливо, ибо страницы одного и того же вектора располагаются в разных блоках одного и того же модуля, а случай, когда происходит формирование страниц вектора в скалярной части, можно не учитывать, так как этот процесс слишком медленный.

3.2. Введение спецпроцессора в векторное ИУ

Для выполнения операций между элементами вектора внутри одного вектора нежелательно передавать страницы этого вектора в скалярную часть. Для этого можно внутри одного векторного процессора построить специализированный вычислитель, выполняющий часто встречающиеся операции между элементами внутри вектора.

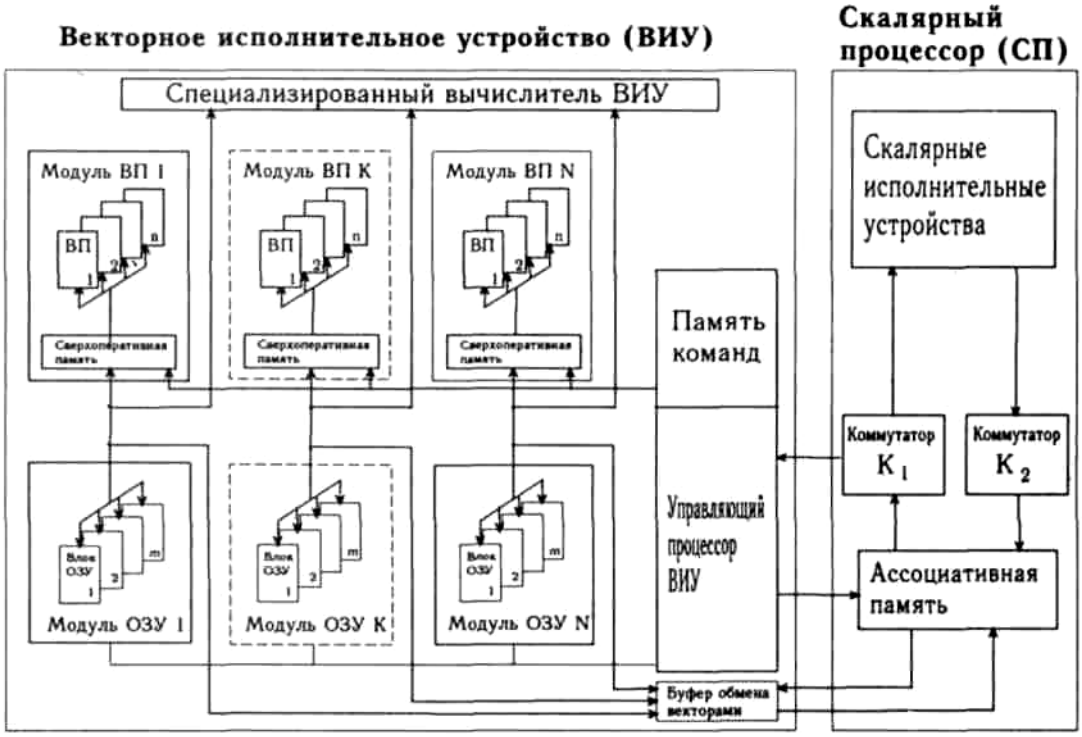


Рис.2, Схема векторного процессора с фрагментами оптимизации его структуры

К таким операциям могут быть отнесены: операция сложения всех элементов страницы, операция перестановки элементов местами как в странице, так и в поле всего вектора, операция исключения элементов из вектора по определенному

правилу и т.д. Специальный процессор может иметь сверхоперативную память на два или три вектора и специальную схему оптимизации этих операций. Так, например, скорость выполнения операции перестановки элементов матрицы может быть увеличена в n^2 раз по сравнению с выполнением ее на традиционной архитектуре [4] (Рис.2).

3.3. Введение сложных векторных операторов (экстракодов)

Следующим естественным шагом оптимизации векторного ИУ является введение экстракодов, определяющих в поле операций целые группы команд, наиболее часто встречающихся в вычислительных процессах. Подпрограммы экстракодов могут храниться в управляющем процессоре и иметь свои коды (экстракоды) операций над векторами. При операции экстракода запускается подпрограмма его выполнения.

Так может быть реализована операция умножения матриц и ряд других сложных операторов. Наличие эскракодов предусматривает введение в каждом векторном микропроцессоре регистров сверхоперативной памяти для хранения промежуточных результатов подпрограммы выполняемого эсктракода. Введение эсктракодов может значительно увеличить скорость выполнения этих операций и, в первую очередь, за счет сокращения обращений к ОЗУ.

3.4. Опережающий просмотр последовательности векторных команд

Следующий прием оптимизации вычислительного процесса векторного ИУ может вызвать существенный пересмотр принципов взаимодействия между векторной и скалярной частью суперЭВМ. Она сводится к опережающему просмотру заявок на выполнение операций, поступающих на векторный процессор. Так, управляющий векторный процессор может, не дожидаясь выполнения операции над пакетом ее дескрипторов, определить место результата этой операции, сформировать дескриптор результата и отослать его в скалярную часть. Продолжая таким образом взаимодействие векторной и скалярной части, можно "вытянуть" из выполняемой подпрограммы всю ближайшую очередь векторных операций. Анализируя их зависимость, можно динамически формировать подпрограммы, сокращая обращения к векторной памяти. Последнее предложение по оптимизации операций в векторном ИУ, безусловно, требует более глубокой проработки. Реализация этих идей, возможно, приведет к введению векторной КЭШ памяти в каждом процессорном модуле.

Заключение

Приведенные соображения по разработке векторного исполнительного устройства говорят о том, что на однопроцессорной суперЭВМ нетрадиционной архитектуры при элементной базе, обеспечивающей темп работы полупроводниковой АП в 10 нс, уже сегодня возможно построить процессор с максимальной производительностью на скалярных операциях 10 Гфлопс (10^{10}), а на векторных более ста Гфлопс (10^{11}). При этом принцип аппаратного распределения ресурсов вычислительных средств будет сохранен, что обеспечит максимальное

автоматическое распараллеливание вычислительных процессов и хорошую загрузку аппаратных средств.

Литература

1. В.С.Бурцев. Система массового параллелизма с автоматическим рапределением аппаратных средств суперЭВМ в процессе решения задачи. Юбилейный сборник трудов ОИВТА РАН, М, 1993, т2, с. 5-27.
2. В.С.Бурцев, В.Б.Федоров. Ассоциативная память на принципах оптической обработки информации для суперЭВМ нового поколения (в данной книге).
3. V.B.Fyodorov. Optoelectronic multiport associative memory for data flow computing architecture. Proc.1994 Int.Conf. on Optical Computing, Inst. Phys. Conf.Ser., 1994, V139. pp87-92.
4. V.S.Burtsev. Application of optical methods of information processing in a supercomputer architecture. International Journal of Optoelectronics, 1994, v9, N6, p. 489-503.

Основные этапы научной деятельности Бурцева В.С.

Бурцев Всеволод Сергеевич, академик Российской Академии наук, директор Института высокопроизводительных вычислительных систем РАН, является крупнейшим специалистом нашей страны в области создания высокопроизводительных вычислительных машин и комплексов как универсальных, так и специализированного применения для управления объектами, работающими в масштабе реального времени.

В.С.Бурцев начал свою деятельность под руководством выдающегося ученого академика С.А.Лебедева еще до окончания Московского энергетического института. Темой его дипломной работы была система управления первой советской быстродействующей электронной машины - БЭСМ АН СССР. Уже на дипломном проектировании он стал одним из ведущих разработчиков в создании этой системы.

Под его научным руководством и при непосредственном участии проведены следующие научные исследования, которые имели большое научно-техническое значение и были внедрены в промышленность.

1953-1956 гг. Бурцев В.С., являясь ответственным исполнителем, разработал принцип селекции и оцифровки радиолокационного сигнала. На основе этого принципа был осуществлен съем данных о цели с радиолокационной станции и ввод их в вычислительную машину. Успешно был проведен эксперимент одновременного сопровождения нескольких целей вычислительной машиной. На базе этой работы была написана кандидатская диссертация, а на защите члены совета единогласно проголосовали за присуждение Бурцеву В.С. степени доктора технических наук. Успешный эксперимент со съемом данных с радиолокационных станций в корне изменил структуру управляющих противоракетных и противосамолетных комплексов.

1956-1961 гг. Под непосредственным руководством В.С.Бурцева разработаны принципы построения вычислительных средств Противоракетной обороны (ПРО) и создан высокопроизводительный вычислительный комплекс для решения задачи высококачественного автоматического управления сложными, разнесенными в пространстве объектами, работающими в масштабе реального времени.

Вычислительные комплексы ПРО были оснащены самой быстродействующей в то время серийной ЭВМ М-40, в которой Бурцевым В.С. впервые были предложены принципы распараллеливания вычислительного процесса за счет аппаратных средств - все основные устройства машины (арифметическое, управления, ОЗУ, управления внешней памятью и т.д.) имели автономные системы управления и работали параллельно во времени. Впервые был использован принцип мультиплексного канала, благодаря которому без замедления вычислительного процесса удалось осуществить прием и выдачу информации с десяти асинхронно работающих направлений с общей пропускной способностью 1 млн. бит/с

1961-1968 гг. Под непосредственным руководством В.С.Бурцева для создания сложных боевых комплексов была разработана первая высокопроизводительная полупроводниковая ЭВМ 5Э92б с повышенной структурной надежностью и достоверностью выдаваемой информации, основанными на полном аппаратном контроле вычислительного процесса. В этой ЭВМ впервые был реализован принцип многопроцессорности, внедрены новые методы управления внешними запоминающими устройствами, позволяющие осуществлять одновременную работу нескольких машин на единую внешнюю память. Все это дало возможность по новому строить вычислительные управляющие и информационные комплексы для систем ПРО, управления космическими объектами, центров контроля космического пространства и так далее. Многомашинные вычислительные комплексы с автоматическим резервированием хорошо зарекомендовали себя в реальной работе.

1969-1972 гг. Бурцев В.С., являясь Главным конструктором, создал первую вычислительную машину третьего поколения для возимой серийной противосамолетной системы С-300. Это была трехпроцессорная ЭВМ, построенная по модульному принципу. Каждый модуль (процессор, память, устройство управления внешними связями) был полностью охвачен аппаратным контролем, благодаря чему осуществлялось автоматическое скользящее резервирование на уровне модулей в случае их отказов и сбоев, практически без прерывания вычислительного процесса. Комплекс, равный по производительности БЭСМ-6, занимал объем не более одного кубического метра. Эти комплексы в системе С-300 и до настоящего времени стоят на боевом дежурстве и продаются в другие страны.

1973-1985 гг. Являясь Главным конструктором многопроцессорного вычислительного комплекса (МВК) "Эльбрус-1" и "Эльбрус-2", Бурцев В.С. наряду с принципиальными схемотехническими вопросами большое внимание уделял конструктивно-технологическим вопросам, принципиальным вопросам системы охлаждения и повышения интеграции элементной базы, а также вопросам автоматизации проектирования. В процессе создания МВК "Эльбрус-2" по его инициативе и при непосредственном участии созданы новые быстродействующие интегральные схемы, высокочастотные групповые разъемы, многокристальные и большие интегральные схемы, микрокабели, прецизионные многослойные печатные платы. Это было большим вкладом в развитие технологии в нашей стране.

1980 г. Были закончены работы по созданию МВК "Эльбрус-1" общей производительностью 15 млн. оп/с.

1985 г. Успешно завершены Государственные испытания десятипроцессорного МВК "Эльбрус-2" производительностью 125 млн. оп/с. Оба комплекса освоены в серийном производстве.

При создании этих комплексов были решены принципиальные вопросы построения универсальных процессоров предельной производительности. Так, динамическое распределение ресурсов сверхоперативной памяти исполнительных устройств и ряд других решений, впервые используемых в схемотехнике, позволили в несколько раз увеличить производительность каждого процессора. С целью дальнейшего повышения производительности комплекса были решены фундаментальные вопросы построения многопроцессорных систем, такие как

исключение взаимного влияния модулей на общую производительность, обеспечение обезличенной работы модулей и их взаимной синхронизации.

1986-1993 гг. Разработана структура суперЭВМ, основанная на новом не фон-Неймановском принципе, обеспечивающая существенное распараллеливание вычислительного процесса на аппаратном уровне. Эта архитектура использует новейшие принципы оптической обработки информации, обладает высокой регулярностью структуры и позволяет достичь производительности 10^{10} - 10^{12} оп/с. Принципиальной особенностью предлагаемой архитектуры является автоматическое динамическое распределение ресурсов вычислительных средств между отдельными процессами и операторами. Решение этой проблемы освобождает человека от необходимости распределения ресурсов при программировании параллельных процессов в многомашинных и многопроцессорных комплексах.

Работы по исследованию и созданию новых архитектур ЭВМ проводились в рамках "Программы Основных направлений фундаментальных исследований и разработок по созданию оптической сверхвысокопроизводительной вычислительной машины Академии наук" (ОСВМ РАН).

В настоящее время В.С.Бурцев является научным руководителем фундаментальных исследований по разработке различных нетрадиционных архитектурных решений высокопроизводительных вычислительных машин с использованием новых физических принципов, а также системного программного обеспечения с целью создания информационно-вычислительных комплексов с максимальной производительностью 10^{12} - 10^{14} оп/с.

Бурцев В.С. удостоен Ленинской и Государственных премий, награжден орденами Ленина, Октябрьской революции, Трудового Красного знамени и медалями. За цикл работ "Теория и практика создания высокопроизводительных многопроцессорных вычислительных машин" ему присуждена премия АН СССР имени С.А.Лебедева.

Он является автором более 150 научных работ, опубликованных как в нашей стране, так и за рубежом, которые положены в основу проектирования новых вычислительных средств и используются в учебных целях в ведущих ВУЗах России.

Бурцев В.С. ведет большую работу по подготовке научных кадров. Под его руководством успешно защитили диссертации на соискание ученой степени кандидата и доктора технических наук более 40 человек. Он преподавал более 20 лет в Московском физико-техническом институте со дня его основания. В настоящее время Бурцев В.С. является заведующим филиала кафедры "Микропроцессорные системы, электроника и электротехника" и научным руководителем кафедры "Высокопроизводительные вычислительные системы" Московского авиационно-технологического университета им. К.Э.Циолковского.

Содержание

В.С. Бурцев

Параллелизм вычислительных процессов и развитие архитектуры суперЭВМ

юбилейный выпуск

Оригинал - макет подготовлен в ИВВС РАН
на персональном компьютере IBM PC

Подписано в печать 22.01.97. Заказ Тираж 500 экз.
Отпечатано в типографии издательства "НЕФТЬ И ГАЗ"