6.6. ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА Cm*

Система Cm* была разработана Университетом Карнеги-Меллона и относилась к микроВС. Целью разработки было создание ВС из достаточно большого числа (до 100) микропроцессоров для исследования аппаратурно-программных решений в области архитектуры вычислительных систем на базе БИС. Существенное внимание было уделено вопросам обеспечения надежности и снижения показателя "стоимость/производительность". При построении ВС были использованы простые микропроцессоры LSI-11 фирмы DEC, совместимые с мини-ЭВМ PDP-11.

*6.6.1. Архитектура микроВС Cm**

Система Cm* (рис.6.6–6.8) в сравнении с системой C.mmp (см. рис.6.2) приобрела заметные архитектурные усовершенствования, в ней полнее были реализованы принципы модели коллектива вычислителей. Архитектура Cm* стала более близкой к архитектуре ВС с программируемой структурой. В самом деле, в микроВС Cm* пара "элементарный процессор, ЭП – локальная память, ЛП" выполняла функции вычислительного модуля (Computer Module), а вычислительный модуль в совокупности либо с контроллером *K* отображения адресов (рис.6.6,а), либо с локальным коммутатором (ЛК, рис.6.6,б) реализовали функции элементарной машины, если пользоваться терминологией систем с программируемой структурой. В первом случае ЭМ выглядила как двухполюсник, а во втором – как многополюсник (не менее двух полюсов). В состав ЭМ могли включаться внешние устройства (со своими контроллерами).

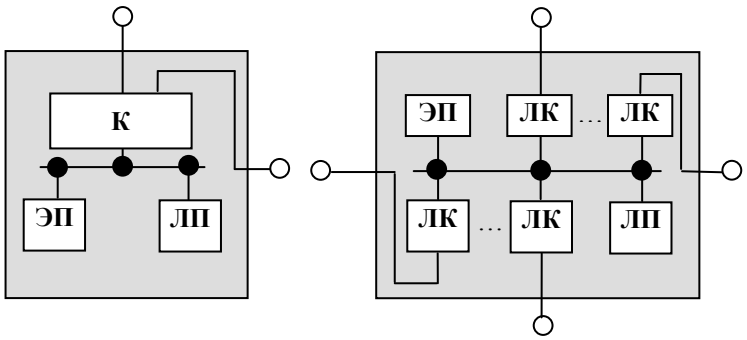

Рис.6.6. Элементарная машина системы Cm*:
а) – на базе контроллера; б) – на базе локального коммутатора

Взаимодействие между вычислительными модулями осуществлялось через "распределенный коммутатор" – сеть связи, образуемую подсоединением контроллеров отображения адресов к межмодульным шинам (рис.6.7) или (и) отождествлением полюсов локальных коммутаторов различных элементарных машин (рис.6.8). Контроллер – это специальный процессор, который выполнял все функции связанные с передачей сообщений. Локальный коммутатор имел простую структуру, которая обеспечивала параллельную передачу слова между процессором.

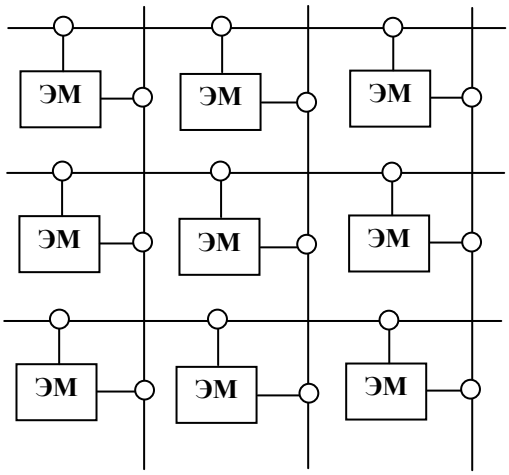

Рис. 6.7. Каноническая структура системы Cm*

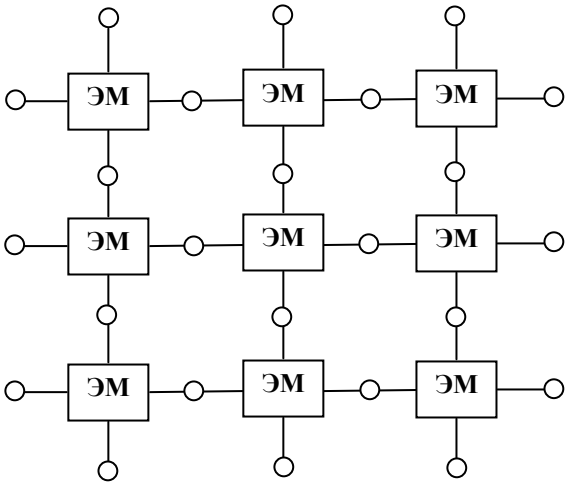

Рис. 6.8. Фрагмент двумерной структуры системы Cm*

Отметим характерные особенности архитектуры вычислительной системы Cm*.

Реконфигуриремость – способность микроВС к априорной адаптации своего состава и структуры сети межмашинных связей к конкретной области применения.

Масштабиремость (наращиваемость) – способность микроВС Cm* к развитию с целью увеличения производительности, объема памяти и полосы пропускания каналов связи. В системе нет принципиальных ограничений на число ЭМ и число связей между ними.

Общедоступность и распределенность памяти. Память системы Cm* состояла из общей памяти и локальной памяти элементарных машин. Вся память системы была потенциально доступна для всех процессоров. За счет наращивания числа ЭМ имелась принципиальная возможность обеспечить превалирование по емкости локальной памяти всех ЭМ сравнительно с общей памятью. В этом смысле можно считать память системы Cm* распределенной.

"Локальность" программы – свойство, положенное в основу архитектуры микроВС Cm*. Эффективное функционирование системы будет достигнуто, если большая часть программы и данных, с которыми работает каждая ЭМ, будет храниться в ее локальной памяти. Для Cm* коэффициент обращения к локальной памяти, определяемый отношением "число обращений к локальной памяти/число обращений к общей памяти", был равен 0,8 …0,9. Ясно, что "локальность" программы будет обеспечена при применении методики крупноблочного распараллеливания сложных задач (см. 3.3).

Эффективность использования ресурсов системы обеспечивалась возможностью организации параллельной обработки исходных данных элементарными машинами и параллельных межмашинных обменов информацией.

Экономичность ВС, т.е. удовлетворительные значения отношения "стоимость/производительность" – результат применения большого числа недорогих серийных микропроцессоров.

Надежность функционирования микроВС достигалась за счет распределенности ее ресурсов. В системе не было критического ресурса, отказ которого приводил бы к отказу ВС в целом. Аппаратурно-программные средства позволяли "удалять" из

структуры неисправные компоненты. Контроль по четности, дистанционное диагностирование и повторение команд обеспечивали обнаружение и исправление не только устойчивых, но и перемежающихся отказов аппаратуры.

Следует заметить, что архитектура системы Cm* допускала формирование структур, например, подобных структуре на рис. 6.8, но в которых в узле устанавливается не простейшая конфигурация элементарной машины (вычислительный модуль с контроллером), а группа ЭМ (как композиция вычислительных модулей и локальных коммутаторов). Связь между ЭМ группы и с контроллером отображения адреса (общим на группу) осуществлялась специальной шиной. Число ЭМ в группе не превышала 14. При этом ограничении на топологию межгрупповой сети связи и на число групп не накладывались. Вычислительная система с такой структурой являлась суперсистемой, в качестве ее элементов вуступали микроВС Cm*. В суперсистеме были потенциально допустимы реализация большого набора функций, адаптация под классы решаемых задач и сферы применения с учетом показателей производительности, надежности и экономической эффективности.

*6.6.2. Средства обеспечения надежности микроВС Cm**

Для достижения надежного функционирования системы Cm* использовались специальные аппаратурно-программные средства. В микроВС Cm* реализован подход, основанный на введении в контроллер отображения адреса специальной аппаратуры – ловушек (hooks). За счет возможности организации взаимодействия этих ловушек с процессором LSI-11 была обеспечена хорошая программная поддержка при разработке системной аппаратуры и ее диагностировании, а также предоставлены удобные условия для отладки микропрограмм на реальной аппаратуре.

Ловушка предоставляла микропроцессору LSI-11 (названному hooks-процессором) возможность тщательного исследования и изменения внутреннего состояния контроллера. Ловушки позволяли загружать в управляющую память процессора микропрограмму, считывать значения сигналов на шинах и управлять генератором тактовых импульсов процессора (выполнять операции “пуск”, “останов”, “один цикл”). Всякий раз при остановке тактового генератора (вследствие либо останова, задаваемого микропрограммой, либо нарушения четности в управляющей памяти или памяти данных) hooks-процессору выдавался сигнал прерывания. Сложность аппаратуры ловушек оценивалась 10% от стоимости процессора LSI-11.

Кроме того, в локальный коммутатор были введены регистры, которые предназначались для хранения информации в ходе диагностирования и восстановления после обнаружения программного и аппаратурного отказов. Почти обо всех отказах, включая отказы, обнаруживаемые контроллером при внешних обращениях, процессор извещался через систему прерывания.

Все внешние обращения к памяти любого вычислительного модуля выполнялись с помощью коммутации сообщений. До получения сигнала подтверждения каждое сообщение хранилось в буфере. По истечении периода ожидания или при получении извещения об ошибке контроллер пытался вновь передать требуемое сообщение, возможно, даже по другому физическому тракту. Пользователь об этом извещялся лишь в том случае, если память, к которой делалось обращение, оказывалась изолированной или находящейся в состоянии устойчивого отказа.

Вся память системы была защищена контролем на четность. При нарушении четности адрес блокировался и поступал соответствующий сигнал в контроллер, с тем чтобы он мог повторить обращение к памяти.

Контроль информированных трактов на четность в сочетании с коммутацией сообщений обеспечивал обнаружение и исправление ошибки в одном разряде. Сообщение

передавалось с вертикальным признаком четности. В месте приема сообщения проверялись как горизонтальный, так и вертикальный признаки четности. Тогда в случае одиночной ошибки в результате пересечения двух признаков четности однозначно идентифицировался разряд, содержащий ошибку.

Установлено, что большинство ошибок в системе составляли ошибки обращения к памяти. Поэтому операционная система была ориентировна на анализ ошибок отмеченного класса.

*6.6.3. Система самодиагностики микроВС Cm**

Контроль и диагностирование свободных от работы вычислительных модулей в Cm* производится системой самодиагностики, представляющей собой последовательность из четырех диагностических программ.

Диагностическая программа для памяти состоит из 13 тестов, в том числе теста “галоп” (gallop test), теста бегущих “0” и “1” (marching ones and zeros) и теста сдвига. Один прогон такой программы для динамической МОП-памяти произвольной выборки емкостью 56К байт занимает приблизительно 13 мин.

Диагностические программы системы команд и системы прерывания предназначены для проверки функционирования микропроцессора LSI-11. Так как эти две программы короткие, то перед переходом к следующей диагностической программе они пропускаются по несколько раз.

Диагностическая программа системной аппаратуры проверяет, во-первых, регистры и тракты данных локального коммутатора, во-вторых, часть контроллера отображения адреса и содержит несколько тестов для проверки отдельных частей памяти.

Система самодиагностики размещается в машине-диспетчере (в мини-ЭВМ PDP-11) на магнитной ленте. Среднее время t° между загрузкой двух очередных диагностических последовательностей в свободные вычислительные модули выбирается экспериментально. На начальном этапе эксплуатации Cm* было взято t°=7 мин. Для такой загрузки используется последовательный канал связи между вычислительными модулями микроЭВМ Cm* и машиной-диспетчером.

Процесс самодиагностики идет непрерывно и заключается в следующем. Сначала машине-диспетчеру сообщается о том, что система самодиагностики берет управление на себя. После этого машина-диспетчер рассматривает вычислительный модуль, реализующий программу самодиагностики, как обычный терминал и позволяет ему “войти” в вычислительную систему (подобно пользователю) и затребовать в машину-диспетчер отчет о состоянии Cm* в целом. В этом отчете указывается, какие вычислительные модули находятся в рабочем состоянии, какие из них свободны от работы. Затем машина-диспетчер по запросу от системы самодиагностики подключается к свободным от работы модулям, загружает в них диагностические программы и подает команду “пуск”. Запросы системы самодиагностики на подключение свободных от работы модулей поступают через t° единиц времени. В случае увеличения потребности в ресурсах микроВС Cm* со стороны пользователей системы самодиагностики отказывается от диагностирования отдельных модулей.

Диагностические программы загружаются поочередно; условием загрузки очередной программы в свободный от работы вычислительный модуль является либо успешное завершение заданного числа прогонов, либо обнаружение отказов после определенного числа прогонов текущей диагностической программы. В последнем случае предпринимается попытка получить от всех диагностических программ максимально возможное количество информации об условиях проявления отказов.

Система самодиагностики способна реагировать на некоторые ситуации автоматически. Одна из таких ситуаций – это зацикливание модуля. Если модуль в течение определенного времени не реагирует, то производится его повторная загрузка и пуск. Вторая ситуация – превышение времени ожидания, фиксируется в том случае, если ожидаемый от машины-диспетчера символ не поступает в течение определенного времени; при этом печатается сообщение о превышении времени, а выполняемая задача ставится в очередь на повторное решение.

Система самодиагностики предоставляет два вида информации. Первый – это извещение об ошибке в том или ином модуле, печатаемое при ее обнаружении и содержащее: обозначение модуля; наименование выполняемой диагностической программы; текущее время, прошедшее после предыдущей ошибки в данном модуле; сведения об ошибке. Второй вид информации – извещение о состоянии вычислительной системы в целом, которое содержит время начала работы системы самодиагностики, время выдачи извещения и сведения о состоянии всех модулей.

Диагностические программы ориентированы на обнаружение устойчивых отказов в Cm*. Однако эти программы (с некоторой вероятностью) способны обнаружить и перемежающиеся отказы. Предложенный способ самодиагностики микроВС Cm*, несмотря на свою гибкость, все же нельзя назвать вполне отвечающим современным требованиям. Если изъян состоит в том, что система самодиагностики работает с помощью машины-диспетчера (что является узким местом) и не на основе взаимного тестирования элементарными машинами друг друга. *Необходима распределенная система самодиагностики.*

*6.6.4. Анализ и модификация архитектуры микроВС Cm**

Статистическая обработка информации по применению системы самодиагностики показала, что средняя наработка на отказ ϑ элементарной машины (состоящей из процессора, динамической МОП-памяти емкостью 48 К байт, локального коммутатора и периферийных устройств) равна ϑ=127,7 ч при t°=7 мин и ϑ=562,6 ч при t°=30 мин. Приведенные значения ϑ следует рассматривать как верхние оценки средней наработки на отказ системы. Установлено также, что одновременное появление перемежающихся отказов в нескольких модулях возможно с вероятностью 0,17, и такие ситуации возникают в среднем через 1000 ч. Отношение интенсивности перемежающихся отказов к интенсивности отказов было равно 100:1 при t°=7 мин и 30:1 при t°=30 мин.

Ясно, что вероятностная модель микроВС Cm* существенно сложнее модели для мини-ВС C.mmp, поэтому разработчиками были подвергнуты анализу лишь конкретные конфигурации системы. Количественный и качественный анализ убедил разработчиков ВС из Университета Карнеги-Меллона в том, что с позиций надежности и живучести архитектура системы Cm* более перспективна, чем архитектура системы C.mmp (и конечно, систем семейства Burroughs).

Главным недостатком структуры системы Cm* явилось использование нераспределенного диспетчера (мини-ЭВМ PDP-11). Отказ мини-ЭВМ PDP-11 приводил к невозможности реализации функций операционной системы и, следовательно, к отказу системы как единого аппаратурно-программного ансамбля. Этот недостаток системы Cm* безусловно можно было устранить. В самом деле, возможности архитектуры системы Cm* значительны: в частности, она допускала такую модификацию, когда исчезал главный недостаток – единый аппаратурный диспетчер. Мини-ЭВМ PDP-11 могла быть исключена из структуры Cm*, а операционная система – реализована в распределенном виде, т.е. ее компоненты могли быть размещены в элементарных машинах. Системное программное

обеспечение Cm* (даже если не использовать машину-диспетчер) могло быть построено на основе стандартного программного обеспечения для PDP-11 и LSI-11.

Описанная модификация архитектуры Cm* могла бы улучшить количественные характеристики микроВС как единого аппаратурно-программного комплекса и, в частности, положительно сказалась бы на надежности и живучести системы. Однако такая модификация существенно приблизила бы системы Университета Карнеги-Меллона к ВС с программируемой структурой, развиваемым не только у нас в стране, но и за рубежом.