Цифровые процессоры обработки сигналов

Витязев С.В. РГРТУ. 2012

4. Основы построения ЦСП — память: архитектура фон Неймана, гарвардская и модифицированная гарвардская архитектуры памяти; кэш-память; иерархическая архитектура памяти

Память в составе архитектуры ЦСП выполняет функции хранения отсчетов входного и выходного сигналов, хранения программных кодов алгоритма обработки, его параметров, массивов промежуточных результатов и других типов данных и программ. Принципы организации памяти и ее взаимодействия с операционным ядром называются **архитектурой памяти** ЦСП. Архитектура памяти определяет общую эффективность сигнального процессора не менее, чем состав его вычислительных блоков. Вычислительные блоки выполняют обработку данных; задача архитектуры памяти — успевать подводить к вычислительным блокам новые данные и снимать результаты обработки. Если новые операнды не будут успевать поступать на вычислительные блоки, то вычислительные ресурсы процессора будут простаивать и не иметь смысла.

Вспомним, что архитектура классического ЦСП исходит из принципа быстрой реализации операции умножения с накоплением. Естественно, архитектура памяти также должна выбираться исходя из данного принципа. На рис. 4.1 проиллюстрировано, какие требования предъявляются к архитектуре памяти при реализации операции умножения с накоплением.

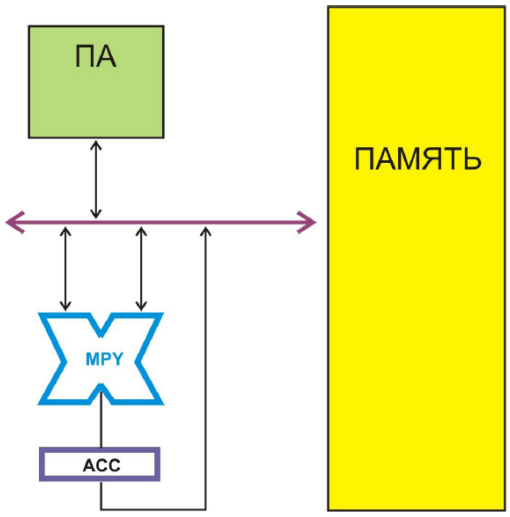

Рис. 4.1

Для реализации одной операции умножения с накоплением необходимо «взять» из памяти два операнда для подачи на умножитель MPY, прочитать команду, декодируемую программным автоматом (ПА), и после умножения и добавления к содержимому регистра-аккумулятора (ACC) записать результат в память. Таким образом, для реализации умножения с накоплением за один такт необходимо в одном такте производить 4 обращения к памяти (4 пересылки между памятью и ядром): два чтения данных, одна запись данных и одно чтение команды. Можно, однако, заметить, что обычно операции умножения с накоплением повторяются многократно и интерес представляет только конечный результат накопления, и записывать в память содержимое регистра-аккумулятора на каждом такте не нужно. Таким образом, мы приходим к необходимости реализовывать 3 операции пересылки данных между ядром и памятью. Итак, архитектура памяти должна строиться так, чтобы обеспечивать 3 пересылки за такт.

Ранее мы говорили, что архитектура сигнального процессора унаследовала общие принципы построения ЭВМ, сформулированные фон Нейманом. По фон Нейману архитектура памяти процессора строится в соответствии со структурой рис. 4.2.

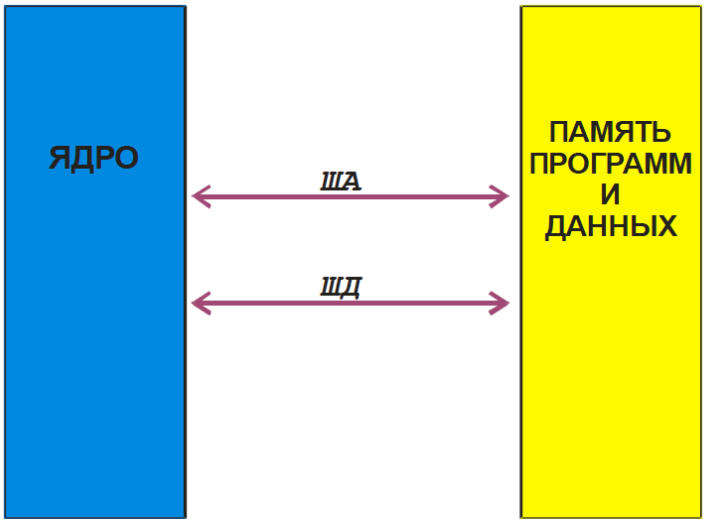

Рис. 4.2

Данная архитектура носит название **фон-неймановской** или **принстонской** (по названию университета, в котором работал Джон фон Нейман). Она подразумевает наличие одного блока памяти для программ и данных и наличие одного канала пересылки программ и данных между ядром и памятью — шины адреса (ША) и шины данных (ШД). **Архитектура фон Неймана** позволяет выполнить только одну пересылку за такт, поэтому она не подходит для применения в сигнальных процессорах. Ее достоинство было в другом — впервые был предложен принцип работы с командами аналогично данным. Команды хранятся в едином адресном пространстве наравне с данными, они извлекаются из памяти

аналогично операндам, программа может полностью изменяться простой перезаписью нового набора команд в память процессора.

Разделение блока памяти на два независимых блока хранения программ и данных привело к появлению **гарвардской архитектуры памяти** (рис. 4.3).

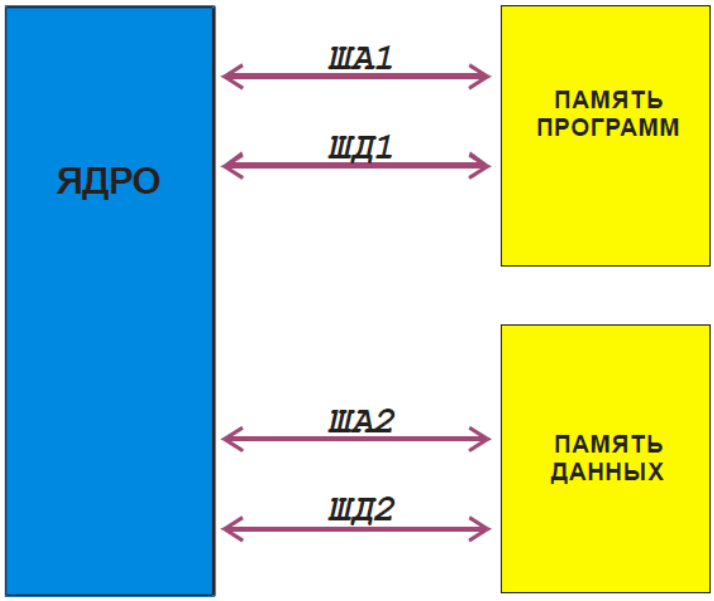

Рис. 4.3.

Гарвардская архитектура обеспечивает два независимых канала пересылки данных и программ между ядром и памятью, однако, этого также оказывается мало для цифровых сигнальных процессоров.

Чтобы обеспечить 3 пересылки между памятью и ядром за такт, в сигнальных процессорах применяются различные модификации гарвардской архитектуры памяти (**модифицированная гарвардская архитектура**).

На рис. 4.4 показана **архитектура памяти с множественным доступом**.

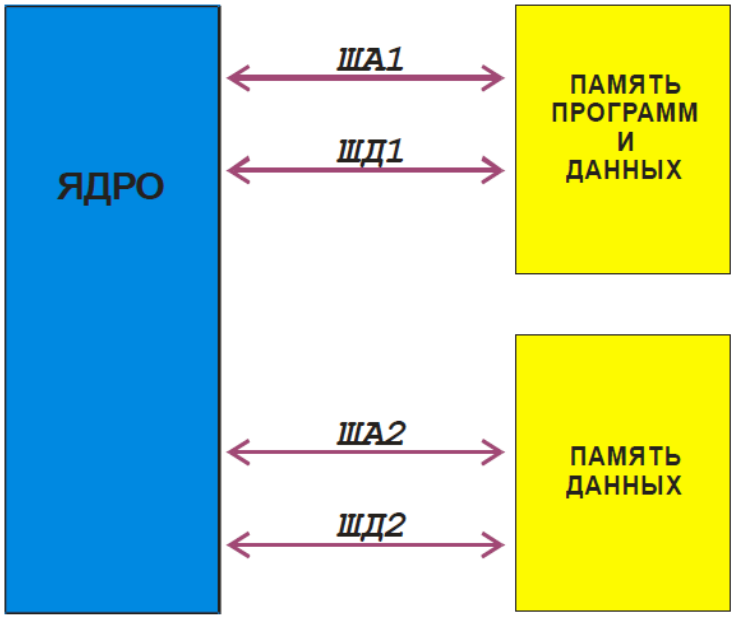

Рис. 4.4.

В такой архитектуре память данных и память программ разделены аналогично гарвардской архитектуре памяти. Имеются два канала параллельной пересылки программ и данных. Однако, в отличие от гарвардской архитектуры, совместно с командами в памяти программ могут храниться и данные, а блок памяти реализован так, что за один такт к нему может быть выполнено два последовательных доступа.

Другой вариант модификации гарвардской архитектуры представляет применение **многопортовой памяти**, когда один из блоков памяти поддерживает два параллельных канала доступа (рис. 4.5).

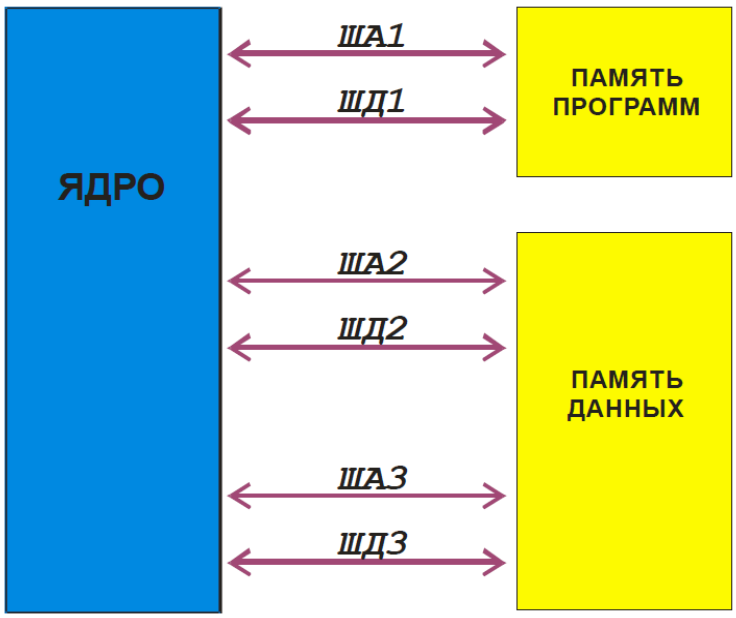

Рис. 4.5

Применение **архитектуры памяти с тремя независимыми блоками памяти** и тремя наборами шин является другой разновидностью архитектуры памяти, применяемой для построения ЦСП (рис. 4.6).

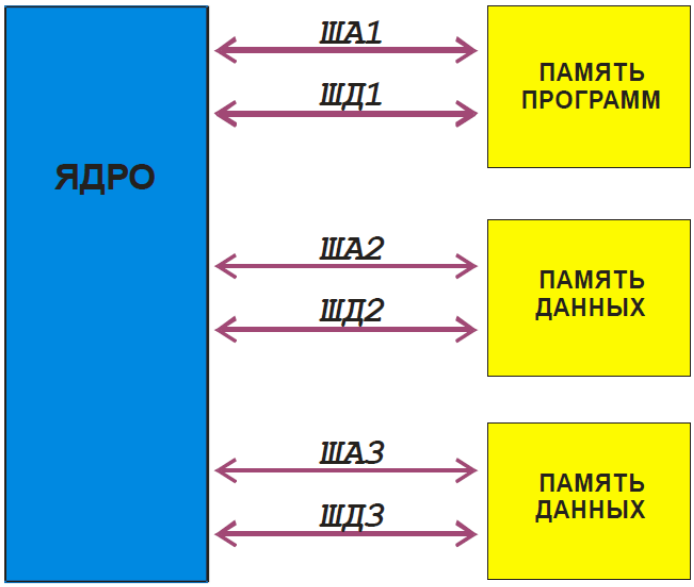

Рис. 4.6.

Оригинальной архитектурой памяти представляется **память с буферизацией блока команд**. Ее применение основано на том, что, как правило, интенсивный вычислительный процесс и пересылка данных соответствуют циклическим структурам программных кодов. Действительно, вспомним КИХ-фильтр. Многократно повторяющаяся операция умножения с накоплением, сопровождающаяся интенсивной пересылкой данных между памятью и вычислительными блоками, соответствует циклу, реализующему операцию свертки. То же самое характерно для большинства типовых алгоритмов ЦОС. Цикл, обычно, — это многократно повторяющийся набор небольшого числа команд. Возникает идея выделения специального небольшого буфера (например, относящегося к программному автомату), в который при первой итерации цикла копируются команды, составляющие тело цикла. Тогда при следующих итерациях цикла чтение команд будет производиться из буфера, освобождая канал обмена с памятью программ и данных для чтения операндов. Развивая идею применения специального буфера хранения наиболее актуальных команд, делая ее более гибкой и универсальной, мы приходим к применению кэш-памяти (рис. 4.7).

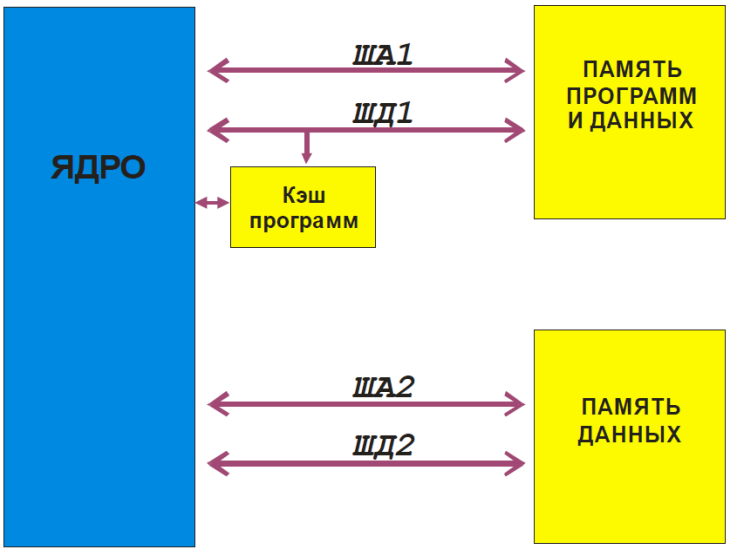

Рис. 4.7.

Кэш-память представляет собой буфер промежуточного хранения команд или данных. Соответственно, различают кэш команд и кэш данных.

Можно выделить две цели, с которыми кэш-память применяется в сигнальных процессорах. Первая — освобождение канала обмена между ядром и памятью для обеспечения 3 параллельных обращений к памяти за такт, о чем говорилось выше. Вторая цель — обеспечение высокой скорости работы с медленной памятью. Рассмотрим выше представленную структуру более подробно и проиллюстрируем с ее помощью принцип работы кэш-памяти.

Процессор работает в обычном режиме, производя чтение команд из памяти программ и поочередно их исполняя. Параллельно с чтением команд из памяти и загрузкой их в программный автомат происходит запись прочитанных команд в кэш-память — локальную память небольшого объема. При следующем обращении к памяти программ, прежде чем прочитать команду, происходит проверка, не находится ли эта команда уже в кэш. Если команда записана в кэш, то ее чтение производится из кэш-памяти по параллельному каналу, а основной канал доступа в память программ (и данных) освобождается для чтения операндов. Очевидно, что данный подход будет эффективен только в случае повторяющегося выполнения одних и тех же команд в цикле, причем первая итерация цикла (когда команды еще не загружены в кэш) будет выполняться дольше. Кэш выступает в качестве промежуточного буфера хранения команд, однако, это не просто буфер. Кэш-память включает дополнительную логику определения, записаны ли в кэш запрашиваемые команды, и механизм замены старых команд новыми.

Другая цель применения кэш-памяти, о которой уже было сказано выше, — это ускорение работы с медленной памятью. Предположим, мы имеем процессор, включающий операционное ядро, внутрипроцессорную память программ ОЗУ ПП и память данных ОЗУ ПД (рис. 4.8). Кроме того, большой объем программных кодов и данных заставляет использовать дополнительно внешнюю микросхему памяти. Внешняя память, обеспечивая большой объем для размещения программ и данных, работает гораздо медленнее внутренней памяти из-за необходимости пересылок между микросхемами.

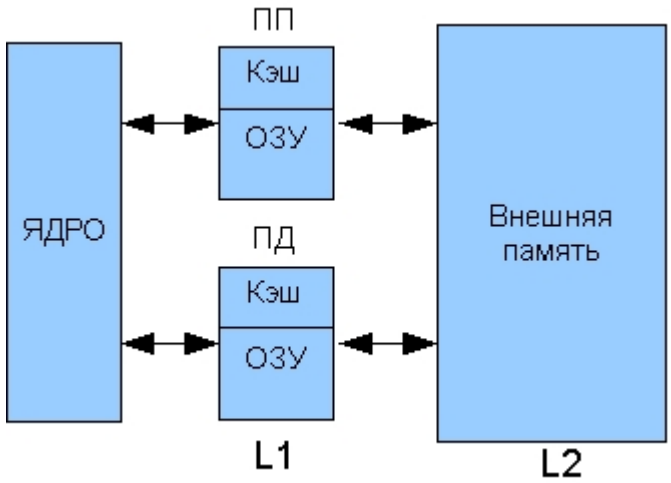

Рис. 4.8.

Чтобы работать с командами и данными, расположенными во внешней памяти, со скоростью ядра процессора, логичным шагом является копирование требуемых в настоящий момент фрагментов кода и данных во внутреннюю память и работа с ней. Например, для КИХ-фильтра перед циклом свертки можно выполнить копирование ядра цикла из внешней памяти во внутреннюю, а также скопировать массивы используемых данных: отсчетов сигнала и коэффициентов фильтра. Реализация цикла будет связана только с внутренней памятью, после чего можно будет загружать из внешне памяти во внутреннюю следующий фрагмент кода и данных. В результате время выполнения будет таким, как если бы мы работали с внутренней памятью, однако, дополнительное время потребуется на предварительное копирование команд и данных из внешней во внутреннюю память.

Механизм кэширования позволяет автоматизировать описанный процесс, сделать его универсальным и минимизировать затраты на перенос данных и команд во внутреннюю память. Принцип работы кэш-памяти остается таким же, как в примере с буферизацией команд. Копирование из внешней памяти в кэш производится всегда. При этом копируется не только текущая команда, но и набор команд (или данных), расположенных рядом с текущей. Вероятность того,

что следующее обращение к внешней памяти будет происходить по соседнему адресу достаточно велика. Поэтому удается предугадать запрос и заранее скопировать нужную информацию в кэш-память.

На рис. 4.8 в блоках внутренней памяти программ и данных выделены области под кэш, используемые для обращений к внешней памяти. Одновременно остаются области обычной памяти ОЗУ, не используемой в качестве кэш. Приведенный пример позволяет ввести понятие иерархической архитектуры памяти.

Иерархическая архитектура памяти предполагает наличие нескольких уровней памяти: **уровень L1, L2, L3** и так далее. При этом к этим уровням памяти может относиться как внутренняя, так и внешняя память. Чем меньше уровень памяти, тем ближе она к ядру процессора в смысле скорости работы с ней, и тем, как правило, меньше ее объем. Чем больше уровень памяти, тем дальше она от ядра — время доступа к ней и ее объем увеличиваются. Для возможности быстрой работы с памятью больших уровней на всех меньших уровнях выделяются области памяти для кэширования данных/программ.

Иерархическая архитектура памяти широко применяется в современных процессорных системах и становится особенно актуальной для многоядерных сигнальных процессоров. На рис. 4.9 показан типовой вариант построения многоядерного ЦСП, включающего три ядра.

Каждое ядро многоядерного ЦСП включает внутреннюю память уровня L1 с разделением памяти программ и памяти данных и выделением областей под кэш. Кроме того, каждое ядро включает внутреннюю память уровня L2 большего объема, объединяющую память программ и данных. Часть этой памяти или весь ее объем может быть использован в качестве кэш для работы с памятью еще более высокого уровня. Память уровня L3 является внешней для ядер ЦСП, но размещается внутри процессора и позволяет организовать взаимодействие между ядрами многоядерного ЦСП. Данную архитектуру можно дополнить внешней памятью уровня L4, большим числом ядер и соответствующих блоков памяти и другими модификациями.

Иерархическая архитектура памяти может применяться наряду с принципами организации взаимодействия между ядром и памятью, описанным в начале данного раздела.

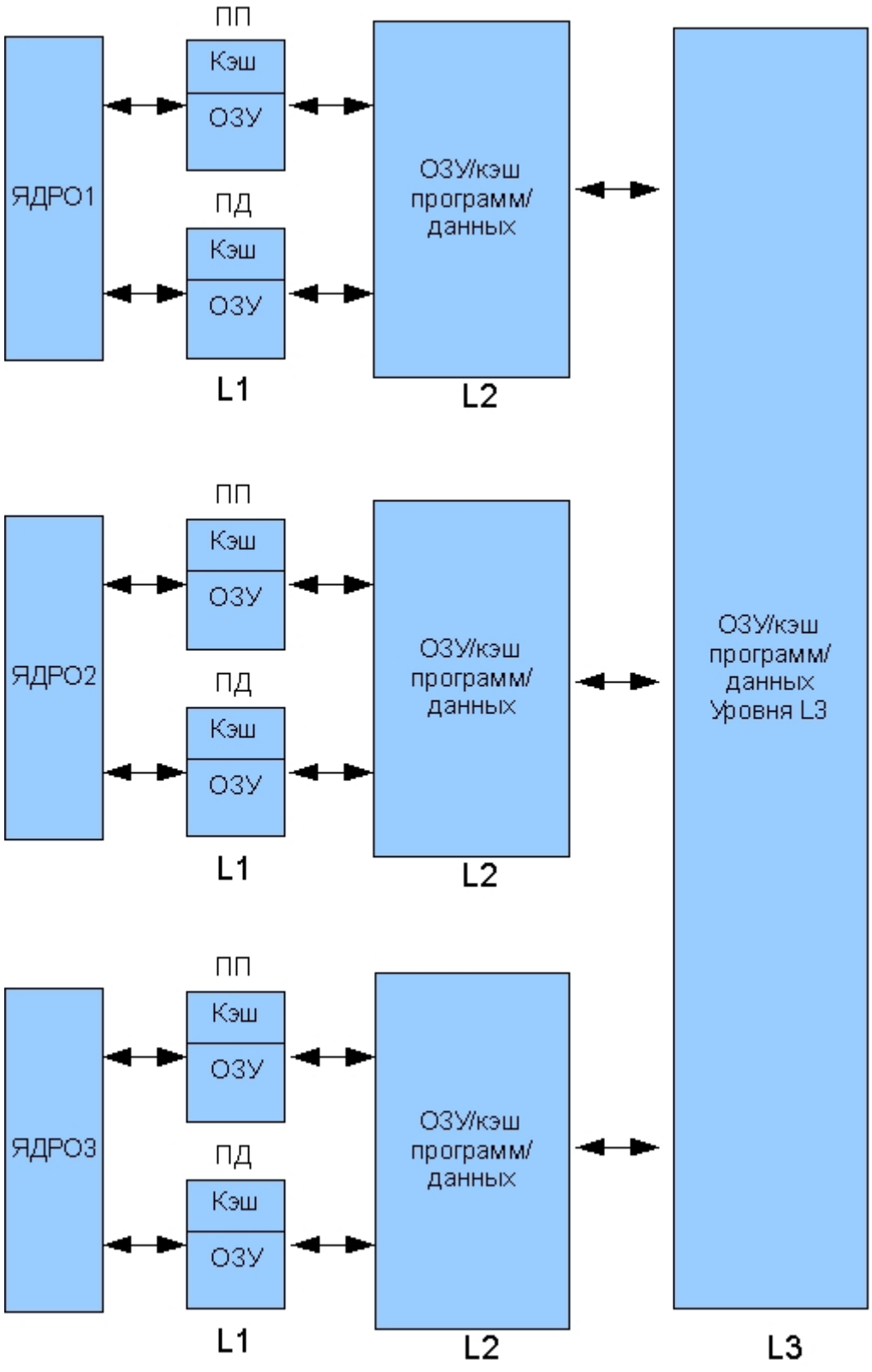

Рис. 4.9.

В заключение отметим следующее. Существенной особенностью архитектур сигнальных процессоров является применение специальных архитектур памяти, обеспечивающих высокую пропускную способность каналов обмена данными между ядром и памятью с применением большого числа шин адреса/данных.

Архитектура памяти часто оказывается «узким местом» процессорной системы, определяющим всю ее производительность. Разработчику системы цифровой обработки сигналов для эффективной реализации разрабатываемого решения необходимо понимать принципы построения архитектур памяти и уметь задействовать ее возможности.

Контрольные вопросы:

1. Для каких целей в составе ЦСП используется память?
2. Почему архитектура памяти имеет большое значение для общей производительности ЦСП?
3. Зарисуйте фон-неймановскую архитектуру памяти.
4. Зарисуйте гарвардскую архитектуру памяти.
5. Чем память с множественным доступом отличается от многопортовой памяти?
6. Что такое кэш-память?
7. Каковы две основные цели применения кэш-памяти?
8. За счет чего наличие кэш памяти позволяет выполнять 3 доступа в память за такт?
9. Каков принцип работы кэш-памяти?
10. В каких случаях примение кэш-памяти не имеет смысла?
11. Зарисуйте иерархическую архитектуру памяти.
12. Что означают уровни памяти L1, L2, L3?