УДК 004.383.3

# ОСОБЕННОСТИ ОБРАБОТКИ СИГНАЛОВ НА ПРОЦЕССОРЕ С РЕКУРРЕНТНО-ДИНАМИЧЕСКОЙ ПАРАДИГМОЙ ВЫЧИСЛЕНИЙ

***А.В. Филин***

## 1. ВВЕДЕНИЕ

Данную работу следует рассматривать как связующую между двумя статьями ([1] и [2]), публикуемыми в настоящем сборнике. Дополнительная информация по рассматриваемой ниже проблеме содержится также в [3].

В [1] даётся представление о возможном характере вычислительного процесса на основе рекуррентно-динамической парадигмы вычислений, обосновывается её принципиальная база как фундамент для построения архитектур вычислительных устройств (на примере рекуррентного векторного процессора обработки сигналов). Там же описывается набор преобразований, которому должны подвергаться алгоритмы обработки сигналов, прежде чем их можно будет выполнять на рекуррентном векторном процессоре (РВП). Общий вывод статьи позволяет надеяться на получение у вычислительных устройств (ВУ), базирующихся на рекуррентно-динамической архитектуре (РДА), новых качественных свойств и функциональных возможностей, улучшенных технико-экономических характеристик, изначально не свойственных компьютерам с классической (фон-неймановской) архитектурой.

В [2] всесторонне рассматривается функционирование операционного уровня РВП, интеллектуальное операционное устройство (РОУ) которого является самодостаточным объектом, способным самостоятельно выполнять все возложенные на него функции, не прибегая к помощи извне, то есть со стороны верхнего (процедурного) уровня архитектуры.

Настоящая статья рассматривает особенности функционально-структурной организации РВП, типы используемых данных и архитектуру процесса вычислений в целом – на всех трёх её уровнях (диспетчерском, процедурном и операционном).

## 2. ОСНОВНЫЕ ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Некоторые понятия, использующиеся в данной статье, определены в [1]**:** "*алгоритм*", "*граф*", "*графодинамика*", "*графодинамический метод*", "*динамическая задача*", "*концепция*", "*парадигма* (в *computer science*)", "*парадигма фон-Неймана*", "*параллелизм*", "*принцип*", "*рекуррентность"*, "*самодостаточность*", "*элемент самодостаточных данных*". Остальные наиболее значимые понятия определяются ниже.

*Архитектура вычислительного устройства* (компьютера) – упорядоченная пара {*ПР, АС*}, где ПР – *принцип работы* устройства, а АС – его *аппаратная структура* (структура аппаратных средств).

- *Принцип работы* (*ПР*) – определяет функциональное поведение (функционирование) вычислительного устройства, специфицируемое его *информационной* и *управляющей* структурами.
- *Аппаратная структура* (*АС*) – представляется структурой *аппаратных ресурсов* (устройств), их *системой межсоединений* и *правилами*, по которым они взаимодействуют.
- *Информационная структура* – представляется набором *типов машинных данных* вычислительного устройства, характеризуемых типом и структурой каждого информационного компонента, представлением их в устройстве (машине), а также операциями преобразования, которые должны выполняться над ними.
- *Управляющая структура* – представляется спецификацией алгоритмов управления, обеспечивающих доступ, интерпретацию и преобразование информационных компонентов вычислительного устройства.
- *Аппаратные ресурсы* – запоминающие, процессорные и другие устройства, в том числе любые программы и микропрограммы, благодаря которым функции вычислительного устройства могут быть приведены в исполнение.
- *Система межсоединений* – объединяет все физические средства, с помощью которых аппаратные ресурсы взаимодействуют (связываются друг с другом).
- *Правила взаимодействий* – управляют коммуникациями и взаимодействием между аппаратными ресурсами структуры, включая и их взаимосинхронизацию.
- *Тип* (*машинных*) *данных* – содержит класс объектов данных и набор функций, применяемых к этим объектам данных.

*Модель* – объект-заместитель, который в определённых условиях может заменять объект-оригинал, воспроизводя выбранные свойства и характеристики

оригинала и предоставляя существенные преимущества удобства (наглядность, обозримость, доступность испытаний, лёгкость оперирования и проч.).

*Система* – совокупность взаимосвязанных объектов (элементов), обособленная от среды и взаимодействующая с ней как *целое*; в силу этого между её объектами существуют определённые связи – отношения; таким образом. закономерности функционирования *системы* определяются не только составляющими её частями, но и общей конституцией, единством..

*Степень параллелизма алгоритма* – количество частей (ветвей) алгоритма, которые могут выполняться одновременно**;** то же, что и степень распараллеливания структуры процесса, достижимая для данного алгоритма.

*Степень параллелизма компьютера* – число процессоров или процессорных элементов, способных работать одновременно.

*Степень параллелизма операционного устройства* – число внутренних *слайсов* (вертикальных каналов обработки данных), способных работать независимо и одновременно.

*Структура* – совокупность отношений между объектами системы, необходимых и достаточных для достижения *цели*.

# 3. ОСОБЕННОСТИ ОРГАНИЗАЦИИ РЕКУРРЕНТНОГО ВЕКТОРНОГО ПРОЦЕССОРА (РВП)

## 3.1. Назначение РВП

РВП предназначается для цифровой обработки сигналов в реальном времени и относится к классу *DSP*-процессоров (*Digital Signal Processor`s*). Поскольку его архитектура ориентируется на производство вычислений рекуррентно представленных алгоритмов ЦОС, она скорее *специализированная*, чем универсальная. Предположительная специализация архитектуры - решение задач ЦОС в низовых звеньях систем управления технологическими процессами, в радиосистемах и радиоустройствах (сотовых телефонах на базе стандартов типа CDMA, TDMA, GSM, D-AMPS и других), а также в составе цифровых портативных аудио- и видеотелефонных аппаратов, то есть всюду, где требуется *высокая скорость обработки сигналов*.

## 3.2. Общая характеристика РВП

Сегодня подавляющее большинство DSP-процессоров не способно (так они спроектированы) самостоятельно выполнять своё предназначение, поэтому управляются от универсального процессора (CPU) того компьютера, в состав которого вводится DSP-процессор. Обычно CPU является ведущим устройством, а DSP – ведомым. В исследуемом (экспериментальном) варианте РВП операционное устройство (а не DSP в целом!) управляется от универсального процессора (работающего в логическом режиме), который также моделирует работу аппаратно не существующих двух верхних уровней архитектуры. Таким образом, готовится платформа для построения полностью рекуррентного РВП (экспериментальный вариант РВП является *частично рекуррентным*). Это основное предназначение *экспериментального* варианта РВП.

Гипотетический вариант компьютерной системы управления (КСУ) некоторым объектом управления (ОУ) с использованием РВП представлен на *рис.1*. Назначение этого рисунка - показать, что любое непосредственное управление объектом (технологическим процессом) практически всегда является *двухконтурным*. Задача *нижнего контура*, соприкасающегося непосредственно с объектом управления (через системы датчиков и исполнительных механизмов) - исполнение в реальном времени режимов контроля, регулирования и управления ОУ с целью поддержания его параметров на заданном уровне. Задача *верхнего контура* (старшего в иерархии, но зависящего от нижнего контура) – проведение оптимизационных расчётов для вычисления значений параметров, определяющих безопасность и экономичность хода технологического процесса, и передача значений этих параметров нижнему контуру для производства операций по обеспечению заданного режима работы ОУ. Наличие двух контуров в КСУ, работающих в разных временных срезах, требует раздельной обработки (без смешивания) двух потоков данных. Первый (сигнальный) поток имеет место между ОУ и РВП нижнего уровня (ввод - обработка “на лету” - вывод), второй – между информационной базой (ИБ) КСУ и РВП

верхнего уровня. Оба потока отображают структуру своих контуров (являются замкнутыми потоками) и тесно взаимосвязаны по данным, управлению и по времени.

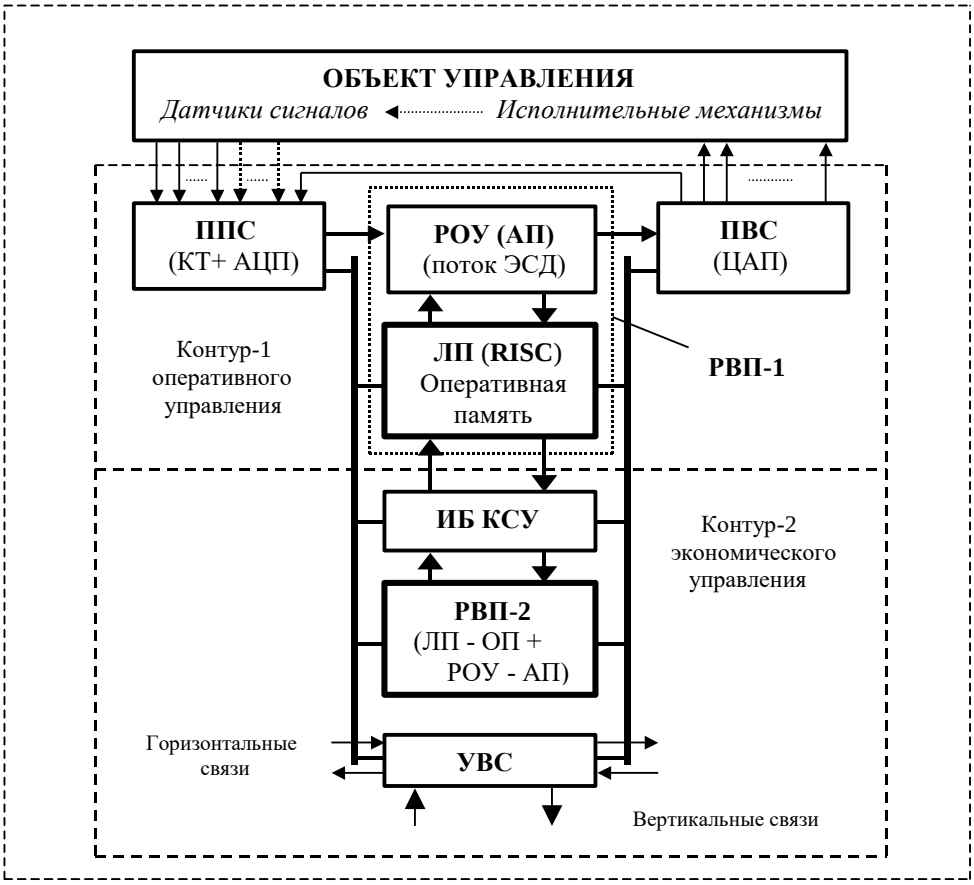


*Рис.1.* Вариант КСУ некоторым объектом с использованием РВП.

*Здесь*: ППС (КТ + АЦП) = подсистема приёма сигналов (коммутатор + аналого-цифровой преобразователь); ПВС (ЦАП) – подсистема выдачи сигналов (цифро-аналоговый преобразователь); УВС – устройство внешних связей; АП – ассоциативная память.

### 3. 3. Состав архитектурных уровней РВП

В принципе РВП может быть выполнено (как логически, так и физически) во множестве вариантов воплощения архитектуры, отвечающих общей концепции, но различающихся конкретными свойствами. Концепция РВП (в задуманном исполнении) естественно-иерархическая и многоуровневая (*рис.2*). На настоящий момент концепция построения РВП предполагает три уровня иерархии, (четвёртый – “ввод-вывод” - считается частью диспетчерского уровня)**:**

а) *диспетчерский* (верхний) уровень;

б) *процедурный* (средний) уровень;

с) *операционный* (нижний) уровень.

Все три уровня в настоящее время изучаются с помощью программных моделей на персональной ЭВМ, играющей роль и логического процессора (модель), и рекуррентного операционного устройства (модель). Выполнение намечавшегося (наиболее экономичного) варианта аппаратной реализации РВП в составе**:** рекуррентное операционное устройство, универсальный RISC-процессор в качестве исполнителя процедурного и диспетчерского уровней (включая и уровень ввода вывода), оперативная память для RISC-процессора - было отложено из-за несостоявшегося финансирования. Аппаратную модель РВП предполагалось подключать к ПЭВМ на уровне подсистемы, управляемой собственным драйвером.

В соответствии с концепцией архитектуры РВП каждый из уровней (подуровней) взаимодействует только с соседними, выше- или нижестоящими, уровнями архитектуры**:** диспетчерский уровень - с верхним подуровнем процедурного уровня и так далее до нижнего подуровня процедурного уровня, который взаимодействует с операционным уровнем. Диспетчерский уровень взаимодействует с внешней средой. В экспериментальном варианте РВП диспетчерский уровень включает в себя три подуровня*:* ввода-вывода, обработки капсул-заданий и обработки капсул-задач.

Основной функцией *диспетчерского* уровня является управление очередностью решения задач, упакованных в пакеты (задания), и их динамическое перераспределение между исполнительными слайсами РОУ, а также реализация интерфейса связи с пользователем и информационной базой (включая компиляцию программ, написанных на ЯВУ). Каждая задача в РВП представляет собой самодостаточную информационную единицу, называемую *капсулой*. Капсула-задание содержит в себе набор имён капсул-задач. Алгоритм решения задачи рекуррентно свёрнут до размеров кода - начального условия развёртки

(НУР) алгоритма [1] – и вместе с начальным операндом упаковывается в одно машинное слово, содержимое которого называется *элементом самодостаточных данных* (ЭСД). Ядром диспетчерского уровня является *программа-диспетчер* (в дальнейшем - "*диспетчер"*), выполняющая также и функцию ввода-вывода. Диспетчер также может быть капсулой, которая саморазворачивается при вызове на исполнение. Структура и механизм исполнения капсул рассматриваются ниже.

Для работы с процедурами (функциями) в архитектуре РВП предусмотрен *процедурный* уровень с числом подуровней, отражающим иерархию процедур и связанных с ними структур данных (аргументов функции и параметров вызова процедуры). Последовательный вызов процедур (функций) ранга *r* отражается в рекуррентной развёртке функциональных полей соответствующих ЭСД на подуровне *r* процедурного уровня архитектуры. При последовательном вызове вложенных процедур (рекурсии) эволюция ВП вовлекает в процесс соответствующее количество подуровней процедурного уровня (*r, r*-1, *r*-2,… ,1) со сквозной рекуррентной развёрткой функциональных полей иерархической структуры данных на всех подуровнях.

Каждый вызов процедуры на подуровне *r* процедурного уровня архитектуры формирует капсулу ранга *r*. Эта капсула включает в себя комплект *статических* наборов данных (НД) с управляющей и вспомогательной информацией, а также требуемое количество содержательных *квазистатических* НД, несущих в содержательной части – аргументы и параметры вызова, а в функциональной части – динамические имена НД, полученные рекуррентной развёрткой при вызове процедуры.

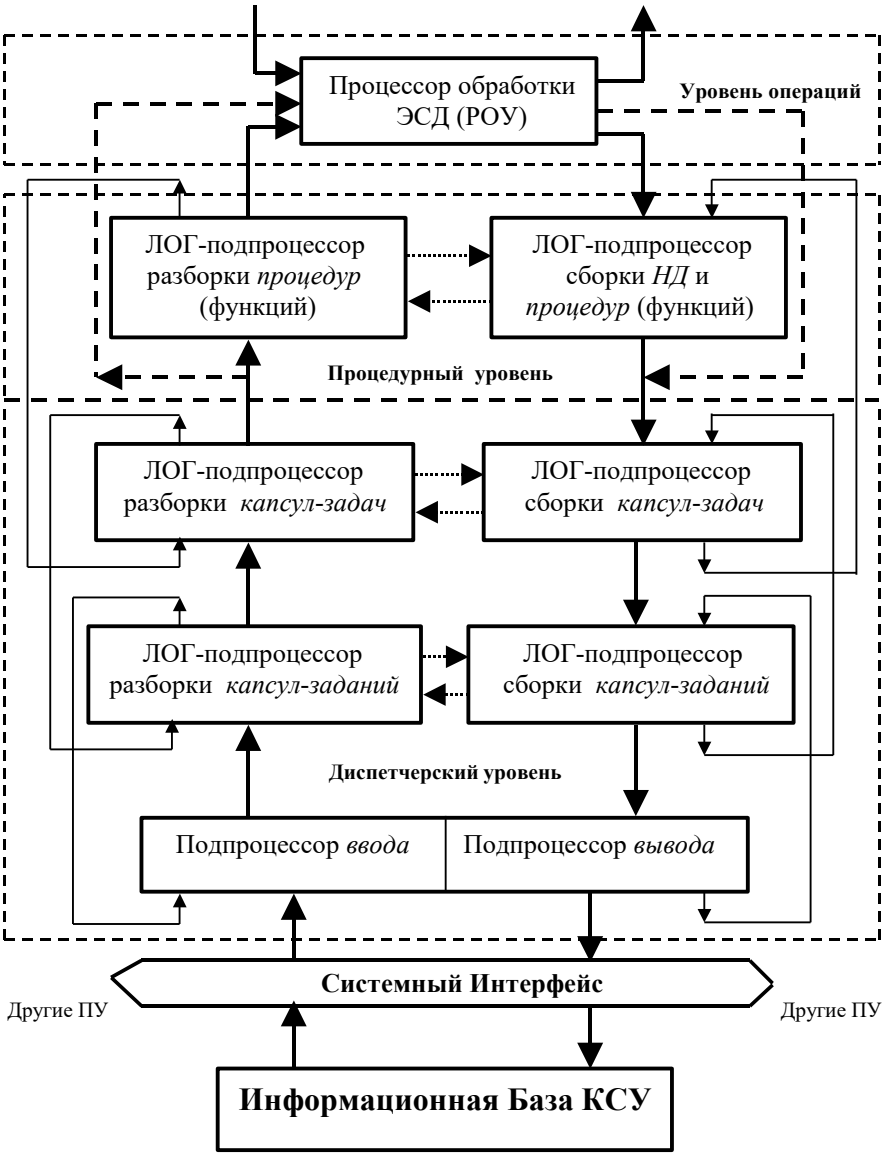


*Рис.2.* Принцип движения потока информации в РВП при взаимодействии его с информационной базой КСУ.
Здесь: ЛОГ – логический; ПУ – периферийное устройство.

### 3.4. Функционально-структурная организация РВП

Ниже рассматривается обобщённая структура РВП, несколько отличающаяся от принятой для проектной разработки. Для уяснения общих принципов организации РВП достаточно считать, что структурно он состоит из двух интеллектуальных устройств – *интеллектуального рекуррентного операционного устройства* и *интеллектуального*

*управляющего устройства* (ИУУ), то есть по своей структурной организации относится к категории *процессоров*. Процессор (в общепринятом определении) есть совокупность операционного (ОУ) и программного управляющего (УУ) устройств, интегрированных в единое целое общностью решаемой задачи, хранящейся в оперативной памяти. В РВП функции ИУУ возложены на *вычислительное устройство*, состоящее из устройств - *логического процессора* (ЛП)*, *устройства оперативной памяти* (УОП) и *канала ввода-вывода,* связанных между собой шинами *чтения* и *записи*. В силу того, что УУ строится на основе процессора, оно названо *интеллектуальным*. УОП вычислительного устройства характеризуется адресным доступом к своим ячейкам, то есть также имеет классическую организацию. Подготовка данных для обработки в РОУ осуществляется, таким образом, программно в вычислителе с помощью ЛП. На логический процессор возложено также программное исполнение функций *процедурного* и *диспетчерского* уровней архитектуры РВП.

* *Примечание.* В нашем случае (для DSP) требуется *логический RISC-процессор, а не классический* (универсальный), поскольку ему для реализации функций верхних уровней архитектуры не нужно выполнять арифметические операции. В нём не должно быть дублирующего арифметического устройства, поскольку уже есть РОУ.

Данные, поступающие на вход РОУ с процедурного уровня, относятся к категории *самодостаточных* – имеющих *при себе* информацию, организующую самопродвижение операндов и их самообработку в пределах оборудования РОУ. Эта информация рекуррентно свёрнута, и *интеллектуальность* аппаратуры РОУ заключается в том, что она умеет её разворачивать, находить второй операнд, выполнять операции по обработке операндов, предписанные данному шагу развёртки, и готовить следующий шаг обработки. И всё это *без помощи извне*.

Структура РОУ имеет *параллельную слайсовую* (вертикальную) *организацию*, теоретически допускающую наращивание числа потоков до 256. Экспериментально проверена в работе на типовых алгоритмах обработки сигналов (на компьютерной модели) структура из 10 слайсов.

Это соответствует десятикратному параллелизму операндов на операциях с однократной точностью.

В каждом РОУ имеются два узла, обрабатывающих операнды**:** традиционное *арифметико-логическое устройство* (АЛУ) и специальное *эвольверное устройство* – (ЭВУ). В задачу ЭВУ (эвольвера) входит развёртка функциональных полей (тегов) слова ЭСД. На вход РОУ с диспетчерского уровня поступает только *один* поток самодостаточных данных, структура которого формируется логическим процессором. Здесь (то есть на входе) поток преобразуется из “вертикального” в “горизонтальный” (разворачивается в вектора по горизонтали) для того, чтобы не отдельные операнды ЭСД поступали на обработку, а сразу обрабатывался *вектор операндов* или его часть (*подвектор*). Поскольку алгоритм содержит в себе параллелизм, последовательный поток самодостаточных данных автоматически превращается в параллельный (множественный) поток. Задействуется сразу столько слайсов РОУ, сколько слов ЭСД (и соответственно, операндов) содержит вектор. В итоге запускается столько вычислительных процессов, сколько допускается на каждом шаге саморазвёртки алгоритма. Таким образом, параллелизм, имеющийся в алгоритме, извлекается полностью за выявленное ранее (при свёртке) число шагов.

### 3.5. Взаимодействие РВП с внешней средой

Специфицируемая архитектура способна общаться с внешним миром с двух полярных концов внутренней иерархии через интерфейсы ввода-вывода**:**

а) *диспетчерского уровня* – со средствами системной поддержки (системными дисками ИБК и операторскими дисплейными консолями);

б) *операционного уровня* - с устройствами связи с объектом управления.

Интерфейс диспетчерского уровня функционирует в категориях архитектурных интерфейсов, то есть наборов данных и капсул.

Интерфейс операционного уровня выполняет основные функции ввода-вывода в РВП. Он же осуществляет преобразования вводимых данных во внутренние форматы данных специфицируемой архитектуры.

Интерфейсы ввода-вывода в экспериментальном варианте РВП предполагаются традиционными для преемственности по периферийным устройствам и контроллерам. В перспективе все устройства и интерфейсы, входящие в РВП, предполагается сделать самосинхронными, обладающими уникальными достоинствами, изложенными в [4, 5].

### 3.6. Механизм образования и использования НУР

Рекуррентная свертка алгоритма выполняется компилятором таким образом, что структура, управляющая его развёрткой, превращается в код НУР этого алгоритма**.** Выходным языком компилятора, таким образом, является язык начальных условий рекуррентной развертки - кодов, размещаемых в словах ЭСД, которые, в свою очередь, собираются в *капсулы* (*рис.3*). После поступления капсулы на вход РОУ и её разборки - представления набора данных в виде вектора, НУР каждого самодостаточного слова капсулы автоматически запускает “программу” динамической развертки, порождающей вычислительный процесс реализации алгоритма, “скрытого” в данном коде НУР. Образующиеся результирующие операнды собираются *в наборы самодостаточных данных* (НД), которые отправляются далее “вверх” на процедурный уровень обработки (для преобразования в *новые капсулы*).

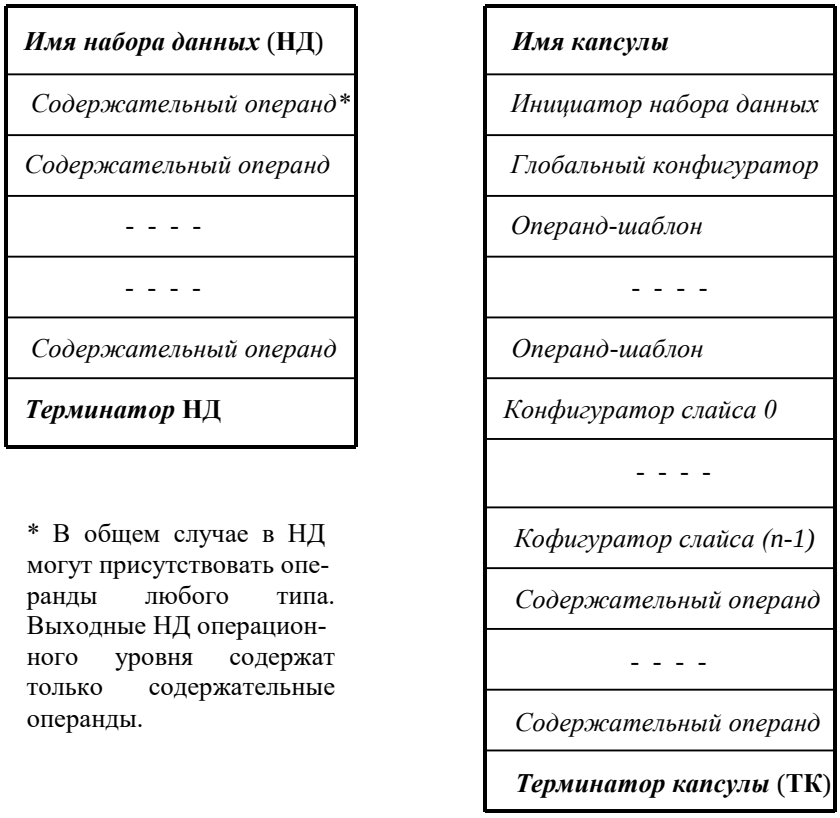


*Рис.3.* Иллюстрация идеи формирования наборов данных и капсул.

## 4. АРХИТЕКТУРА ВЫЧИСЛЕНИЙ НА РВП

### 4.1. Управление задачами

Напомним, что под *задачей* понимается капсула диспетчерского (*Д*) ранга (совокупность оболочки капсулы и НД). Диспетчер выделяет ресурсы (требующиеся для выполнения капсулы), отслеживает процесс разворачивания (исполнения) капсулы на всех этапах, начиная с момента ввода *задания* (набора имён капсул *Д*-ранга) на выполнение одной или нескольких задач до получения результата (сборки новой капсулы-задачи и, возможно, новой капсулы-задания).

Когда необходимо запустить на исполнение несколько капсул одновременно, диспетчер идентифицирует эти задачи (капсулы) и распределяет между ними ресурсы в соответствии с их приоритетами.

Задача, то есть конкретная капсула, соответствующая ей, может обратиться к другой задаче с требованием - выполнить для неё некоторые функции. Задача, которая осуществляет вызов другой задачи, называется *вызывающей*, а задача, к которой произошло обращение, - *вызываемой*. Вызывающая и вызываемая задачи могут поменяться ролями. Так организуется цепочка связей задач, которая отображается в специальных управляющих капсулах-таблицах. Управление задачами заключается в построении графа связей и анализе управляющих капсул с целью принятия решения, какой из задач (капсул) передать управление. Если несколько задач претендуют на один и тот же ресурс в одно и то же время, учитывается их приоритет. Процесс подготовки капсул к исполнению – это процесс программирования задач. Он имеет свои особенности.

### 4.2. Общая организация процесса вычислений

Процесс вычислений в РВП инициируется начальной капсулой (НК), которая разворачивается наподобие куклы-матрёшки и спускает на процедурный уровень первый комплект капсул более низкого порядка (ранга). Начальной капсулой может быть капсула-задание, содержащая набор пакета имён капсул-задач по какой-либо проблеме.

На *первом* этапе вычислений вынимаемые из НК капсулы низшего порядка спускаются до операционного уровня, где начинаются собственно процессы вычислений (производства операций над операндами) и формирования результирующих наборов данных.

На *втором* этапе работы РОУ навстречу нисходящему потоку капсул начинает подниматься поток НД с результатами, инициирующий новые инкапсуляции и подпитывающий нисходящий поток капсул. Этот динамический процесс продолжается до тех пор, пока не иссякнут наборы данных, готовые к инкапсуляции, и система перейдёт в состояние активного ожидания новой начальной капсулы.

Операционный уровень способен также осуществлять ввод-вывод данных. С точки зрения ВП он способен как поглощать (для обработки

или вывода), так и порождать (результатами обработки или вводом) наборы данных. Выводимые НД покидают процесс и способствуют его завершению, а вводимые – подпитывают и активизируют.

### 4.3. Особенности “программирования” задач для РВП

“Программа” в РВП (как системе с рекуррентным внутренним языком) представляет собой**:**

а) некоторое множество исходных кодов НУР в функциональных полях ЭСД, которое будет подвержено рекуррентной развёртке на различных уровнях и подуровнях архитектуры;

б) соответствующее множество правил развёртки этих исходных кодов НУР.

В РДА программирование некоторой задачи сводится к созданию начальной капсулы высшего ранга, однозначно и полно описывающей решение данной задачи. Вся необходимая для её решения информация должна быть инкапсулирована. Информация, инкапсулированная компилятором в начальную капсулу, должна содержать “программу” обработки и все необходимые для задачи исходные данные. Процесс компиляции превращается в “свёртку” исходного алгоритма, изначально представленного в некоторой форме (традиционной - формула, граф, язык – Си, Фортран и т.д., или нетрадиционной), во множество функциональных НУР и правил их развёртки, а затем их объединение с исходными данными в начальной капсуле.

## 5. ТИПЫ ДАННЫХ В РВП

### 5.1. Классификация типов данных

Архитектуре РВП органически присуще иерархическое строение. Количество уровней иерархии архитектуры точно соответствует числу уровней иерархии обрабатываемых машинных *типов данных* (ТД). Каждый тип данных обрабатывается на соответствующем ему архитектурном уровне (операционном, процедурном, диспетчерском, ввода-вывода) и не зависит от способа обработки (аппаратного или

программного). Организация каждого уровня архитектуры предопределена теми типами данных (элементами обработки), которые он должен идентифицировать, расшифровать и обработать. Все упорядоченные по вертикали (иерархически) ТД связаны между собой принципом рекуррентной вложенности – низший тип данных (операционный уровень обработки) является составной частью соприкасающегося с ним старшего типа данных.

Предусматриваются (в той части, которая моделирует РВП) три основных класса типов данных:

а) *элементарный тип данных***:** одиночные скалярные, одиночные входные и одиночные выходные самодостаточные данные;

б) *групповые типы данных***:**

- наборы из одиночных скалярных данных, поступающие с выхода РОУ (без полей),

- наборы из одиночных элементов самодостаточных данных (с заполненными функциональными полями),

- горсти линейных векторов из одиночных элементов самодостаточных данных, поступающие на вход РОУ;

в) *структурированные типы данных***:**

- различные типы записей (с информацией и программами моделируемых уровней РВП),

- элементарные капсулы (базовые оболочки капсул – пустые, без обрабатываемых данных),

- функциональные капсулы (библиотечные прикладные капсулы-задания и капсулы-задачи, полностью готовые к использованию).

Особенность этой классификации в том, что все классы типов данных связаны в систему *свойствами иерархии и рекуррентной вложенности*. В экспериментальном варианте РВП существует три уровня иерархии данных, выстроенных по вертикали (четвёртый – уровень организации файловой системы записей на диск – тот же, что используется в ПЭВМ). Здесь, под *уровнем иерархии* понимается слой

иерархической структуры, всем элементам которого присвоен один номер уровня иерархии. Уровни взаимодействуют между собой с помощью *связи* - механизма взаимодействия, устанавливающего отношения между ними. В РВП определяющей является *вертикальная* связь между данными, находящимися на разных уровнях иерархической структуры. *Связь "по данным"* - это форма связи, при которой обеспечивается передача *аргументов* (фактических данных и их параметров) между уровнями. *Горизонтальные* связи - связи между данными, находящимися на одном уровне иерархической структуры - характеризуют *степень параллелизма*, определяемого размерностью вектора, который состоит из слов ЭСД, поступающих одновременно на обработку. *Связь "по управлению"* между уровнями обеспечивает передачу и возврат управления, а также сохранение и восстановление использованных ресурсов (регистров).

Помимо названных видов связей, существуют еще *функциональные связи* - связи между уровнями (и любыми их компонентами) по *выполняемым функциям*.

Модель РВП оперирует со следующими типами данных, распределенными по уровням следующим образом**:**

***На операционном уровне*** обрабатывается только один класс данных - э*лементарные самодостаточные данные*, структура слова которых задаётся формулой**:**

[слово ЭСД = ***a****-операнд* (***p****-операнд*) **+** *контекст*],

где**:**

- ***a****-операнд* (*операнд активный*) - часть слова интегрированного данного (ЭСД);

- ***p****-операнд* (*операнд пассивный*) - часть активного операнда (число, литерал, константа);

- *операнд* (то же, что и ***p****-операнд*) - основной обрабатываемый элемент скалярного процессора, на котором производится моделирование.

***На процедурном уровне*** обрабатывается класс групповых типов данных**:**

а) наборы слов ЭСД, в которых содержатся только ***р****-операнды* (остальные функциональные поля в слове не заполнены)*;

* *Примечание*. Слова такого набора могут оперативно подаваться сразу же с выхода РОУ на вход РОУ для дальнейшей обработки; при этом на место пустых функциональных полей подставляются поля из находящегося на текущей обработке слова самодостаточных данных.

б) наборы из слов ЭСД (с полностью заполненными контекстными полями);

в) горсть линейных векторов из слов ЭСД, поступающих на вход рекуррентного операционного устройства.

***На диспетчерском уровне*** используются**:**

а) *пустая капсула* (оболочка капсулы без обрабатываемых данных);

б) *функциональная капсула* (библиотечный элемент), содержащая оболочку с набором ЭСД и другую информацию.

Более подробно о содержании капсул и их разновидностях см. в [2]. Отметим только требования к капсулам и тот факт, что программирование в терминах РВП – это программирование на уровне капсул и программирование собственно капсул. Капсулы подразделяются на три группы**:** *служебные* (они же – системные, организующие работу РВП), *внесистемные* (организующие работу с внешней средой) и *пользовательские* (проблемно-ориентированные). Библиотека последних должна быть обширной, чтобы увеличить число областей применения РВП.

В перспективе предполагается разработать технологию и инструментальные средства автоматизированного проектирования систем (библиотек) иерархически вложенных капсул.

### 5.2. Пояснения к типам данных

Под термином “*контекст*” понимается текущее содержимое функциональных полей (в словах ЭСД), содержащих информацию,

которая определяет, как должен обрабатываться ***p****-операнд*, расположенный в этом же слове.

Под ***p****-операндом* понимается объект, который поступает на вход РОУ и далее – на один из входов арифметико-логического устройства (АЛУ) для обработки**;** характеризуется, в общем случае, многообразием представлений**:** *целое без знака, целое со знаком, число с фиксированной запятой, число с плавающей запятой, логическая переменная* и т.д. Перемещается управляющей структурой либо аппаратурой. Обрабатывается как неделимое целое на операционном уровне, поскольку имеет элементарную структуру. Когда ***p****-операнд* размещен в слове ЭСД, он становится *непосредственным* операндом.

Под ***a****-операндом* понимается более сложный компонент слова самодостаточных данных. На операционном уровне он представлен несколькими *типами* (разновидностями) [2]. Типы ***a****-операндов* различаются структурой - наличием в их форматах разных полей: *содержательного, функционального* либо *вспомогательного* назначения.

Каждая капсула собирается из пары стандартных компонент**:** *оболочки* и *ядра* (*набора данных и служебных структур*)**.** *Оболочка* также может быть составной. В общем случае она может состоять из двух частей: *шаблона-заголовка* (паспорта капсулы, содержащего функциональное имя капсулы, время создания, адрес местоположения в памяти, тип шрифта, используемого для описания данных и т.д.) и *шаблона настройки ресурсов* (паспорта с параметрами настройки аппаратуры, в том числе ЭВУ), необходимых для исполнения капсулы. *Набор данных* (в общем случае) представляет собой структуру, также собирающуюся из двух частей: *шаблона набора данных* и собственно *набора данных*. Количество типов шаблонов и их структуры определяются числом типов обрабатываемых данных и наборов данных, характером размещения и особенностями процессов ввода и вывода информации.

Эти компоненты - ядро и оболочка (и их содержимое) - образуют внутреннюю реализацию капсулы. В экспериментальном варианте РВП, где капсула используется только в пределах его программной среды и аппаратуры РОУ, её организационная структура существенно упрощена.

Использование капсулы не требует знания всех подробностей её реализации (“инкапсуляция” – сокрытие подробностей). Достаточно включить имя капсулы в прикладную программу пользователя и обеспечить все параметры, которые требуются для её работы, и процесс программирования закончен. Например, для выполнения операции над конкретным набором данных программист должен будет применить в прикладной программе (капсула) программу рассылки (капсула), обеспечивающую посылку (по сети, слайсам) сообщения объекту (тоже может быть капсулой), который может выполнить эту операцию. Программист не должен определять операцию или писать для нее программный код в программе пользователя.

*Технология капсульного программирования* (ТКП) использует понятия *класса* и *наследования*, используемые в объектно-ориентированном программировании (ООП). Понятие “капсула” (в случае ТКП) принципиально отличается от понятия “объект” (в случае ООП) содержимым капсулы, принятым (рекуррентным) способом представления алгоритма внутри неё и схемой его исполнения.

Понятие *класса* позволяет группировать капсулы, имеющие общие свойства. Программисты могут создавать новые капсулы, задавая наследование существующих капсул (объектов), добавляя новые свойства в существующий класс капсул или переписывая функции наследуемого свойства. Эти требования также предполагается реализовать в капсулах.

Набор классов капсул, отвечающий за конкретное подмножество функций системы, например, за графический интерфейс пользователя, называется *библиотекой классов*. Это означает, что библиотека классов

может строиться не только по вертикали, но и по горизонтали (класс уровня архитектуры - все капсулы, обслуживающие этот уровень).

### 5. 3. Структура слова ЭСД и начало его обработки

Основные типы используемых в РВП данных являются самодостаточными единицами (*рис.4*). На операционном уровне таковым является слово ЭСД, представляющее собой *непосредственное данное* - *аргумент* (или, интегрально, цепочку данных - *Аргумент*), сопровождаемое кортежем *подфункций* (интегрально – *Функцией*), обеспечивающим ему свойство самодостаточности. Функциональная часть слова ЭСД несёт в себе в закодированном виде процедуру обработки аргумента. В эту процедуру входят**:** селекция недостающего аргумента, операция обработки в АЛУ, правило преобразования подфункций в ЭВУ (выполнения шага развёртки рекуррентно свёрнутого алгоритма), управление перемещениями по аппаратной среде и другие действия. Основанием для запуска ВП (на операционном уровне) является событие, под которым понимается момент поступления *слова* ЭСД на обработку. Структура функции (состоящей из подфункций) в каждом слове ЭСД описывает правило динамического развёртывания ВП из его рекуррентной записи на данном шаге существования этого слова. Аргументы (операнды) используются этим правилом для выполнения предписанных операций над ними и воспроизведения тем самым одномоментных графов предварительно свёрнутого граф-алгоритма задачи.

Самодостаточные данные требуют минимума памяти для своего хранения, хотя внутри себя могут скрывать пространные алгоритмы сложной иерархической природы. По времени существования они могут быть статическими, квазистатическими и динамическими.

Структуры форматов слов ЭСД приведены на *рис.2*. Поясним для полноты представления о форматах содержание функциональной части слова ЭСД-3. *Подполе селекции* операнда участвует в операции *паросочетания* – нахождения в ассоциативной памяти операционного

уровня второго слова ЭСД, имеющего такой же код в таком же поле. Факт совпадения содержимого этих полей - это событие, которое заставляет начать выполнение операции в АЛУ. Вид операции задается содержимым подполя *операции*. Функция продвижения данных возложена на подполе *пересылки*. Полная расшифровка структуры слов ЭСД сделана в [2].

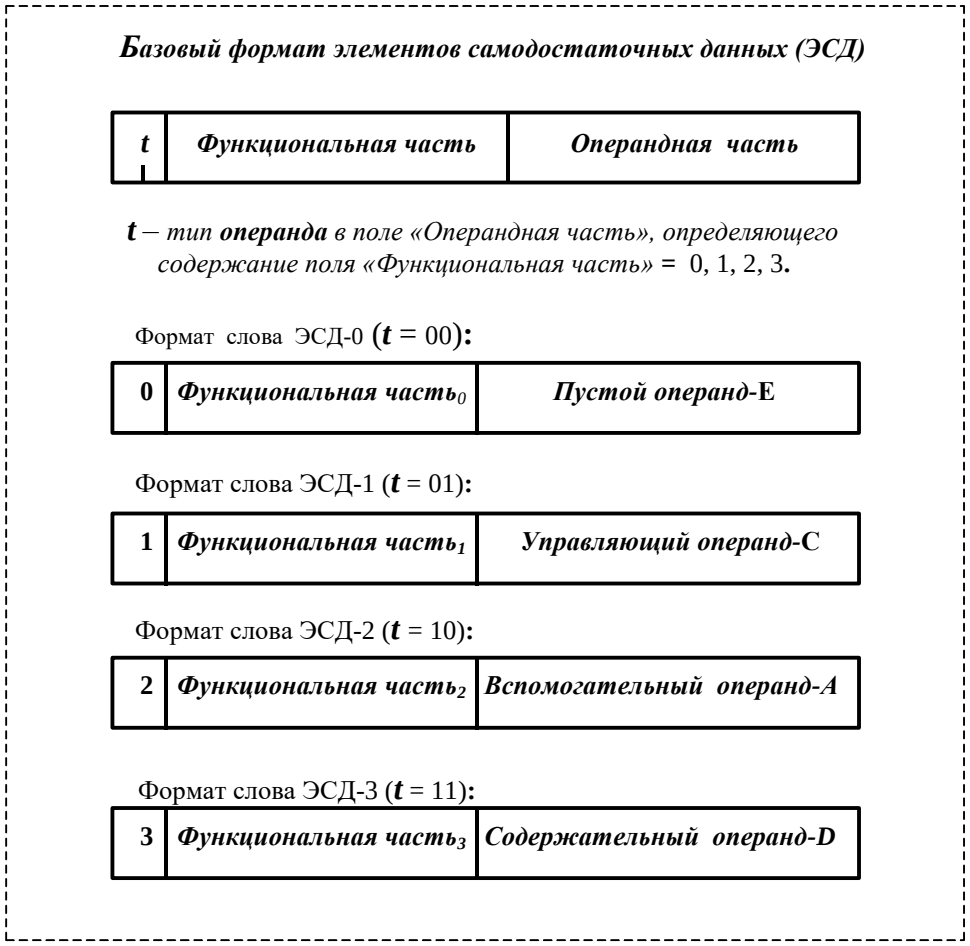


*Рис.4*. Базовые форматы слов элементов самодостаточных данных

### 5. 4. Самосборка вычислительного процесса

Поскольку каждое слово ЭСД содержит в себе (в свёрнутом виде) всю информацию, необходимую и достаточную для обработки операндов на каждом шаге вычислительного процесса, оно полностью определяет и следующий шаг обработки. В результате выполнения шага

обработки появляется следующий ЭСД (новый операнд с новым содержимым функциональных полей), который также определяет следующий шаг обработки. Чтобы такая обработка состоялась, алгоритм закодирован как функция значения начального операнда и правила преобразования функционального поля, сопровождающих этот операнд. Последовательность обработки формируется рекуррентно, и каждый шаг определяется как функция предыдущего состояния. Трасса процесса при этом является не причиной (как в классических и *Data flow*-компьютерах), а следствием осуществления обработки.

Как видим, структура вычислительного процесса формируется по принципу *самосборки* из отдельных шагов обработки. Точнее, процесс самосборки ВП есть *инверсия процесса саморазборки капсулы* конкретного ранга. Аналогично, по состоянию полей инструкций происходит селекция (поиск операнда-партнера) и объединение отдельных скалярных операндов в вектора для выполнения совместных действий. Методом самосборки формируются и более сложные, чем операнд, структуры данных**:** наборы данных, отправляемые РОУ на верхний (процедурный) уровень рекуррентного процессора, капсулы различного ранга и т.д.

## 6. ОРГАНИЗАЦИЯ ВЫЧИСЛЕНИЙ НА ОПЕРАЦИОННОМ УРОВНЕ

### 6.1. Движение операндов внутри РОУ

Процедурный уровень генерирует для нижнего (операционного) уровня два типа самодостаточных информационных структур, образующих поток ЭСД**:**

- собственно *структуры данных* (содержательные операнды)**;**
- *управляющие структуры* (управляющие, вспомогательные и пустой операнды)**.**

Оба типа структур называются одинаково - *операндами* потому, что они обрабатываются не в устройстве управления РВП (как обычно это делается в процессорах CPU или DSP типа), а непосредственно на аппаратуре РОУ. Это и позволяет отнести их к категории *операндов* и

говорить, что на нижнем уровне РВП имеет место только *один* информационный поток активных элементов – поток из слов ЭСД.

На операционном уровне РВП потоки в секциях представляют собой пошаговую развёртку слов ЭСД, начинающуюся от *исходного слова* (содержащего начальный код НУР) *данных* и завершающуюся результатом. Развёртка затрагивает только саморазворачивающиеся части структур ЭСД. В частности, развёртка содержательных частей отражает последовательность промежуточных результатов. Она носит *подчинённый* характер и определяется также и эволюцией функциональных полей в управляющих самодостаточных структурах.

### 8. 2. Последовательность операций обработки ЭСД в РОУ

Процесс обработки потока ЭСД в РОУ складывается из множества подпроцессов их обработки на множестве слайсов. В свою очередь, подпроцесс представляет собой непрерывную последовательность смен состояний функциональных, а иногда и содержательных частей одного слова ЭСД.

На вход РОУ поступает вектор, размерность которого определяется числом слов ЭСД. Одновременно с операндом (операнды рекуррентно не обрабатываются) на обработку поступает и код НУР, но не в традиционное АЛУ, как в случае с операндом, а в ЭВУ. Последовательность обработки складывается из следующих тактов.

*Первый* такт обработки состоит в поиске второго операнда, который должен составить пару с первым в бинарной операции (унарная операция имитируется как бинарная), начинающейся, как только найден второй операнд. Поиск основан на сравнении содержимого специальных призначных (теговых) полей, сопровождающих операнды.

*Второй* такт состоит в выполнении операции над парой операндов и над НУР. Итогом работы АЛУ и ЭВУ являются новый операнд (результат операции) и новое условие развертки. Этой информации достаточно, чтобы интегрировать новый элемент - новое слово ЭСД, если алгоритм завершил работу.

*Третий* такт заключается в сборке (интегрировании выходного операнда с выходным НУР) и записи в память такого интегрированного слова данных. Процесс повторяется до завершения развертки.

Сборка слов, равно как и сборка *набора данных* из этих слов, осуществляется на процедурном уровне (но эти такты к операционному уровню уже не относятся).

Наконец, следует отметить, что непрерывный характер рекуррентной “развёртки – свёртки” алгоритмов обеспечивается без каких-либо внешних принудительных воздействий. Алгоритмы саморазворачиваются как рекуррентные последовательности смены состояний данных и их функциональных полей ЭСД. Чем больше тактов развертки у такой последовательности, тем реже требуется перенастраивать РОУ под следующее слово ЭСД. Это сокращает временные потери и повышает эффективность обработки.

## 9. Особенностей построения вычислительных устройств ЦОС на основе РДП

Учитывая нетрадиционность подхода к построению вычислительных устройств ЦОС на основе рекуррентно-динамической парадигмы вычислений и трудности его усвоения, имеет смысл проследить цепочку наиболее значимых решений, которые были приняты в процессе проектирования логической архитектуры ВУ ЦОС на основе РДП.

1. Исходный (математический) алгоритма задачи должен быть преобразован в особый вид (отличающийся от подобного преобразования, свойственного фон-неймановским и прочим компьютерам)**:** самодостаточный объект, называемый *капсулой–задачей*.

2. Процедура подготовки капсулы к исполнению – процесс программирования задачи, сводящийся к созданию начальной капсулы высшего ранга, однозначно и полно описывающей процесс решения этой задачи. Вся необходимая для решения задачи информация должна быть

“загружена” в капсулу, после чего она приобретает свойство *самодостаточности*. Для пользователя основной единицей программирования является *пользовательская капсула-задача*.

Капсулы ранжированы в соответствии с иерархией архитектуры. Ранг капсулы определяется уровнем архитектуры, на котором капсула создаётся. Теоретически допустима “матрёшка” капсул, связанных свойством рекуррентной вложенности. Процесс её “развёртки-свёртки” можно наглядно представить в виде, например, такой семиуровневой структуры, которую можно интерпретировать по-разному**:**

“***Гигакапсула*** ⇄ мегакапсулы ⇄ килокапсулы ⇄ **капсулы** ⇄ *милликапсулы* ⇄ *микрокапсулы* ⇄ ***пикокапсулы***”

Например, эти семь уровней вложенности капсул можно рассматривать как семь уровней *единой* (гигакапсула всегда одна!) *библиотеки капсул*.

3. На диспетчерском уровне основными единицами обработки являются капсулы-задачи. В состав типовой капсулы входят *программа* (в рекуррентно-свёрнутом виде – код НУР), *набор исходных данных*, *ряд служебных слов*, определяющих *имя капсулы*, *имя набора выходных данных* и *константы*, задающие функции преобразования ФП в ЭВУ. Используемые данные и связи между ними замкнуты внутри капсулы.

4. На операционном уровне основные единицы обработки - *элементы самодостаточных данных* (ЭСД), извлекаемые из капсулы-задачи процедурным уровнем ВУ. ЭСД несёт в себе информацию, достаточную для его самообработки на текущем шаге вычислительного процесса, то есть он является *процессообразующим* элементом капсулы. Формат слова ЭСД имеет единую структуру, содержащую одну из четырёх возможных категорий внутренних частей**:**

а) *содержательную*, несущую обрабатываемые данные (аргументы);

б) *функциональную*, определяющую динамику поведения и преобразования аргументов;

в) *управляющую*, организующую общий ход процесса вычислений;

с) *вспомогательную,* контролирующую внешнюю среду процесса.

5. Рекуррентная саморазвёртка вычислительного процесса – наиболее существенная черта ВУ на основе РДП. Она – следствие саморазвёртки содержательной и функциональной частей ЭСД. Развёртка *содержательных* частей отражает последовательность промежуточных результатов и имеет подчинённый характер, поскольку определяется развёрткой функциональных полей. Развёртка функциональных полей ЭСД является результатом саморазвёртки полей его функциональной части. Каждый шаг развёртки (реализации графов ГЗО и ГР) сопровождается образованием нового результата - ЭСД с новым содержимым функциональной и содержательной частей, несущих исчерпывающую информацию для выполнения следующего шага обработки, на котором новый результат выступает уже в качестве исходных данных.

6. Набор капсул, отвечающих за конкретное множество функций системы, образует *библиотеку капсул данного класса функций*. Напрашивается создание *единой библиотеки,* состоящей из иерархически упорядоченной системы проблемно-ориентированных библиотек.

7. Для облегчения программирования требуется создание технологии *капсульно-ориентированного программирования* (КОП). Основу её должны составить входной языка компилятора, собственно компилятор и инструментальная программная система моделирования и отладки процедур рекуррентного преобразования алгоритмов.

8. Архитектурно-структурной организации ВУ на основе РДП свойственна иерархичность. Общее количество уровней иерархии определяется числом и иерархией типов данных, которые должны идентифицироваться и обрабатываться аппаратурой соответствующих уровней. Все типы данных должны быть взаимосвязаны рекуррентной вложенностью. Число уровней, как и число подуровней внутри каждого уровня, ограничивается только числом типов данных, приписанных данному уровню.

9. Каждая секция ВУ на основе РДП является всего лишь двухканальной аппаратной средой, на которой естественным образом ритмически развиваются процессы “развёртки – свёртки”, имеющие дуалистический характер. Количество слайсов в ВУ (в экспериментальном варианте РВП – количество слайсов РОУ) однозначно задаёт максимально возможную степень распараллеливания алгоритма.

10. Исполнительными элементами РОУ являются пара устройств-преобразователей**:** АЛУ и ЭВУ (другие устройства описаны в [2]). Функция АЛУ – традиционная. Функция ЭВУ – реализация программы пошаговой обработки аргументов алгоритма по мере его рекуррентного саморазвёртывания.

11. Исходные принципы организации ВП на основе РДП не навязывают требования централизованного командного управления, что обеспечивает поддержку любых виды параллелизма, вытекающих из структуры алгоритмов, и тем самым способствует созданию распределённых и параллельных систем ЦОС. Аппаратура операционного уровня представляет собой особую среду, обеспечивающую выполнение операций паросочетания операндов с целью выполнения над ними операционных функций. Размерность (число слайсов) этой среды является единственным ограничителем параллелизма.

## 10. ВЫВОДЫ

Предварительные исследования показали, что вычислительные устройства класса ЦОС (DSP), создаваемые на основе рекуррентно-динамической парадигмы вычислений, могут быть технически реализованы. Никаких специальных требований к своей реализации они не предъявляют. Есть основания утверждать, что они будут превосходить по ряду характеристик аналогичные ВУ этого класса. В числе таких характеристик на первом месте стоит возможность сокращения в десятки и сотни раз объёмов оперативной и постоянной

памяти для хранения программ. Для систем реального времени, работающих в повышенной помехообразующей среде, это чрезвычайно важное свойство, поскольку память является самым ненадёжным элементом современных компьютеров.

Необходимо проведение дальнейших исследований по РДА на основе РДП. Ближайшими задачами следует считать**:** сопоставление РДА с коммерческими и перспективными архитектурами, оценка интегральной производительности и “цены” аппаратной реализации подобной архитектуры, теоретическое обоснование рекуррентно-динамического подхода.

## СПИСОК ЛИТЕРАТУРЫ

1. *Мизин И А., Филин А.В.* Самосинхронизация – естественная основа архитектуры параллельных компьютеров. // Системы и средства информатики. Вып.9. – М.: Наука, 1999. – с. 225 – 241.

2. *Филин А.В.* Динамический подход к выбору архитектуры вычислительных устройств обработки сигналов. // Наст. сб.

3. *Степченков Ю.А., Филин А.В.* Рекуррентное операционное устройство для процессоров обработки сигналов. // Наст. сб.

4. *Филин А.В.* Самосинхронизация – естественный путь обеспечения долгоживучести интегральных схем. // Системы и средства информатики. Вып. 9. – М.: Наука, 1995. - С. 242 – 247.

5. *Филин А.В., Степченков Ю.А.* Компьютеры без синхронизации. // Системы и средства информатики. Вып. 9. – М.: Наука, 1995. - С. 247 – 261.