УДК 004.383.3
УДК 681.324-192

ДИНАМИЧЕСКИЙ ПОДХОД К ВЫБОРУ АРХИТЕКТУРЫ ВЫЧИСЛИТЕЛЬНЫХ УСТРОЙСТВ ОБРАБОТКИ СИГНАЛОВ

А.В. Филин

1. ВВЕДЕНИЕ

Целью настоящей работы является продолжение описания основ выбора и отличительных особенностей архитектуры естественно-надёжных компьютеров (ЕНК), концепция, парадигма вычислений и ключевые принципы построения которой частично изложены в [1]. Представленная там архитектура, называемая в настоящей работе *рекуррентно-динамической архитектурой* (РДА), опирается на одноимённую парадигму управления вычислительным процессом, предусматривающую заблаговременное преобразование алгоритмов задач обработки сигналов к виду, пригодному для исполнения в ЕНК. Но в [1] были рассмотрены только три ключевых принципа - *рекуррентности, самосинхронизации* и *параллелизма*, берущих начало в самой сути *рекуррентно-динамической парадигмы* (РДП) вычислений. Однако рассматривались они там обобщённо, без привязки к какому-либо конкретному вычислительному устройству, поскольку предполагалось, что в перспективе они будут основой архитектуры каждого вычислительного устройства и каждого образца (модели) ЕНК. Ниже рассматриваются все принципы, в особенности не нашедшие освещения в [1], но уже не обобщённо, а применительно к архитектуре конкретного вычислительного устройства – *рекуррентного векторного процессора* (РВП), основное предназначение которого – *цифровая обработка сигналов* (ЦОС) в реальном времени. Из-за нехватки ресурсов проверку диапазона значимости новой парадигмы пришлось

сузить до размеров проекта процессора цифровой обработки сигналов, являющегося структурным элементом ЕНК.

В связи с привязкой принципов к конкретному вычислительному устройству (в данном случае, к РВП), а также для получения общей картины в целом, уточним особенности *рекуррентно-динамического подхода* к проектированию вычислительных устройств (ВУ) ЕНК, определяющего их показатели качества. Поскольку в основу подхода к разработке архитектуры РВП положена новая (не фон-неймановская) вычислительная парадигма, определяющая, во многом, её свойства и характеристики, начнём с рассмотрения её сути. Но сначала определимся с терминологией.

2. ОСНОВНЫЕ ТЕРМИНЫ И ИХ ОПРЕДЕЛЕНИЯ

Чтобы избежать неоднозначностей в трактовке терминов, используемых в настоящей работе, последуем совету Вольтера: «*Прежде чем начать спор – договоримся о терминологии*».

Алгоритм – последовательность правил (план) решения некоторой задачи за конечное число шагов.

Граф алгоритма (задачи) – алгоритм (задачи), представленный в виде графа. Множество вершин V, связи между которыми определены множеством рёбер E, называются *графом* и обозначаются $G = (V, E)$.

Графодинамика – совокупность методов описания и изучения динамических задач*), в которых развитие событий связано с изменением представляющих их графов, то есть с процессами, где во времени меняется либо сам алгоритм решения, либо структура задачи.

Графодинамический метод (описания задач) – дискретный метод, сводящийся к рассмотрению совокупности объектов с позиций графодинамики, существенные свойства которых описываются связями между ними в особых точках графовой траектории. При этом рассматриваемые объекты изображаются в местах появления этих точек на графовой траектории «кружочками», называемыми *вершинами*, а связи между ними – линиями (произвольной конфигурации), называемыми *рёбрами*.

Динамическая задача – задача, структура которой может меняться во время использования; при этом объектом исследования и реализации служит граф в целом, который представляет собой графовую траекторию.

Концепция (от лат. «*conceptio*» – восприятие) – система взглядов по тому или иному вопросу, проблеме или явлению, достаточно полно, целостно и всесторонне раскрывающая их сущность, содержание и особенности**;** ядром концепции является определённая *парадигма*.

Парадигма (в *computer science*) – математическая модель вычислений и принципы построения и использования программно-управляемого вычислительного устройства (компьютера).

Парадигма фон Неймана – модель вычислений, предложенная в 1946 году американским учёным *Джоном фон Нейманом*, ставшая впоследствии теоретической базой создания компьютеров. За небольшим исключением все современные компьютеры являются *фон-неймановскими*. В рамках этой парадигмы им же сформулированы (и апробированы на практике в компьютере *IAS*) *принципы* построения компьютеров, использующих вышеназванную модель вычислений.

Параллелизм – термин, применяемый к независимым (не связанным между собой) процессам, выполняемым одновременно. Различают *статический*, *динамический* и *статико-динамический* виды параллелизма. Параллелизм рассматривается как метод повышения производительности компьютера, основанный на увеличении в нём числа центров обработки данных.

- *Статический параллелизм* – процесс преобразования исходного (последовательного) алгоритма в параллельный (путём выявления частей, способных выполняться одновременно), выявляемый всегда заблаговременно – до начала программирования и исполнения.
- *Динамический параллелизм* – процесс преобразования исходного (последовательного) алгоритма в параллельный, при котором выявление частей, способных выполняться одновременно, осуществляется в ходе его исполнения. При этом стратегия дальнейшего развития процесса – перехода к следующему шагу – уточняется на каждом текущем шаге, в результате чего процесс развёртывается рекуррентно. Этот вид параллелизма также может выявляться заблаговременно и отображаться в алгоритмах.
- *Статико-динамический параллелизм* – основывается на процессах, имеющих место при статическом и динамическом распараллеливании алгоритмов.

Принцип – соглашения, синтезированные на основе наблюдений и экспериментально выявленных закономерностей и несводимые к уже известным принципам**;** каждый принцип, соотносимый с конкретным объектом или процессом, как правило, поддерживается набором требований качественного и количественного типа.

Рекуррентность (рекуррентная вложенность) – фундаментальное структурное свойство активных (самодостаточных) объектов, выражающееся в том, что любой такой объект**:**

- состоит только из множества *подобных* ему объектов младшего ранга;
- всегда является составляющим элементом подобного объекта старшего ранга (подчиняется свойству *иерархической упорядоченности*);
- характеризуется *возвратностью*, то есть повторяемостью всех свойств объектов (структурных и функциональных) на любых уровнях вложенности.

Самодостаточность – свойство активного объекта (технической или биологической системы), обеспечиваемое наличием в нём всего необходимого для реализации логичного и эффективного поведения в определённых (собственных) целях.

Элемент самодостаточных данных – это машинное слово, состоящее из двух частей (полей) - «*Функция*» и «А*ргумент*». Аргумент сопровождается кортежем подфункций (интегрально – *Функция*), обеспечивающих ему свойство самодостаточности. В функциональной части слова рекуррентным способом закодирована процедура (последовательность шагов) обработки аргумента.

* *Примечание*. Здесь и во всех других определениях термин «задача» может быть заменён любым из следующих терминов - «объект», «система», «подсистема» и т.д. - без искажения смысла.

3. РЕКУРРЕНТНО-ДИНАМИЧЕСКАЯ ПАРАДИГМА ВЫЧИСЛЕНИЙ

Теоретической базой рекуррентно-динамического подхода к организации процесса вычислений алгоритмов реального времени является *теория рекуррентно-динамических графов*. Пусть $\boldsymbol{x(t)}$ – граф, существующий в момент $\boldsymbol{t;}$ тогда закон изменения графа во времени может быть записан в форме некоторого рекуррентного динамического процесса**:** $\boldsymbol{x(t+1)=F[x(t)]}$, $\boldsymbol{t = 0, 1, 2, \ldots ,}$ (1)
где $\boldsymbol{F}$ – некоторый оператор, преобразующий граф, исполняемый в момент времени $\boldsymbol{t}$ в граф, который появится в момент времени $\boldsymbol{t+1}$.

Если назвать $\boldsymbol{x(0)}$ начальным графом, то последовательность графов $\boldsymbol{x(t),}$ возникающая из $\boldsymbol{x(0)}$ в соответствии с (1), и будет *графовой траекторией*. Фактически выражение (1) олицетворяет собой задачу («программу»), алгоритм которой был заблаговременно представлен в виде последовательности одномоментных графов, а затем рекуррентно свёрнут в «точку» $\boldsymbol{x(0)}$, код которой представляет собой начальное условие его саморазвёртки. Как видим, в рекуррентно свёрнутой «программе», представляющей собой рекуррентно свёрнутый алгоритм обработки набора (потока) операндов, описывается не поток инструкций, а поток *правил*, порождающих шаги обработки операндов.

Сам же поток интегрально представляет собой последовательность слов *самодостаточных данных* и *самодостаточных управляющих структур* (их несколько типов), каждое из которых обязательно несёт в себе (вплоть до входа в операционное устройство), помимо одного операнда, *начальное условие саморазвёртки* граф-алгоритма.

Исполнение начального условия порождает последовательность одномоментных сменяющихся графов $x(1),x(2),\ldots,x(t)$, образуя *рекуррентно-динамический* вычислительный процесс. Очевидно, что при рекуррентной свёртке достигается мощное сжатие алгоритма и, соответственно, существенная экономия памяти – самой ответственной (и наименее надёжной) компоненты современного компьютера. Поскольку архитекторы фон-неймановских компьютеров сегодня весьма озабочены проблемой «*укрощения терабайт*», предлагаемая парадигма способствует решению этой проблемы наилучшим образом.

Очевидно, что новая парадигма вычислений, как, впрочем, и любая другая, определяет не только общую организацию компьютера, но и все его составляющие – *данные* и *управляющие структуры*, теоретическую базу архитектуры, структуру и взаимосвязи потоков данных, звеньев вычислительного процесса, аппаратного и программного обеспечения в РВП и любой модели ЕНК.

Поскольку процедуры описания алгоритмов с помощью динамических графов и последующая их рекуррентная свёртка требуют квалификации, разрабатываются соответствующие программы для их автоматизации.

4. ЦЕЛЕВЫЕ УСТАНОВКИ НА РАЗРАБОТКУ АРХИТЕКТУРЫ ВЫЧИСЛИТЕЛЬНОГО УСТРОЙСТВА ДЛЯ ЦОС

4.1. Главная целевая установка

Следует подчеркнуть, что главная цель разработки РВП состояла в выполнении двух (основных) задач**:**

- в поиске *более эффективной архитектуры процессора обработки сигналов* (в международной транскрипции DSP–процессора, от *Digital Signal Processor*) по сравнению с известными архитектурами;

- в тщательной *отработке архитектурных методов повышения эффективности вычислений в параллельной системе,* главным образом за счет снижения накладных расходов не только на сам вычислительный процесс (ВП) и обеспечивающие его ресурсы, но и на отыскание такого параллелизма, который можно было бы назвать *естественным*.

Концепция обработки сигналов, парадигма и принципы организации архитектуры РВП выбирались, исходя из этих предпосылок. Другими установками, которые способствовали успешному поиску эффективной архитектуры, были**:**

а) *допустимость проблемной ориентации архитектуры*;

б) *необходимость достижения высокого уровня новизны* (патентоспособности) *архитектуры и изделий на её основе*;

в) *обеспечение возможности её дальнейшего развития*.

4.1.1. Проблемная ориентация архитектуры

Проблемная ориентация и возможности архитектуры определяются, во многом, возможностями парадигмы вычислений. Суть новой парадигмы (как показано выше) - в преобразовании статических алгоритмов в динамические посредством специального механизма их предварительной рекуррентной свёртки (компрессии) и последующей саморазвёртки (декомпрессии) в процессе исполнения. Саморазвёртка (динамическое исполнение) алгоритма инициируется поступлением на обработку очередного слова с *элементом самодостаточных данных* (ЭСД), обладающим средствами поддержки вычислительного процесса за счёт собственных ресурсов, а не извне. Допускается иметь в устройствах более одного потока ЭСД, что характерно для алгоритмов ЦОС. Вычислительный процесс обладает внутренней способностью к самосинхронизации в точках соприкосновения пар операндов по мере рекуррентного саморазвёртывания алгоритма. Таким образом,

предлагаемая архитектура ориентирована на решение математических задач, алгоритмы которых могут быть представлены *рекуррентно-динамическими графами* (РДГ), обладающими внутренней способностью к операциям «свёртки – развёртки». Алгоритмы ЦОС и изображений, имеющиеся в большом количестве в анналах дискретной математики, в полной мере удовлетворяют этому требованию.

4.1.2. Степень новизны в принятии архитектурных решений

Создаваемая архитектура РВП относится к числу *нетрадиционных* (не фон-неймановских), поскольку базируется на нетрадиционной парадигме вычислений. Нетрадиционность архитектуры, прежде всего, заключается в том, что управление ходом вычислительного процесса (ВП) на операционном уровне осуществляется с помощью элементов (слов) единственного потока данных, каждый из которых является *самодостаточным* образованием. Характерной особенностью этих элементов является наличие в них функциональных полей, определяющих действия на текущем и следующем шагах ВП. Но они используются не для инициации выполнения инструкций обработки операндов, как это имеет место в известных *Data flow*-компьютерах, также управляемых потоком данных, а непосредственно для управления развёрткой содержимого функциональных полей, сопровождающих данные, и обработки операндов по мере их готовности. В результате этого количество фаз обработки слов ЭСД уменьшается до предельно-возможного минимума – до трёх: «*сравнение функциональных полей – выполнение операции – запись результата*» [2]. Операции в потоковом операционном устройстве РВП выполняются только при наличии всех операндов (данных) для их выполнения. В функциональной части слова ЭСД в закодированном виде хранится «программа», представляющая собой рекуррентно свёрнутый алгоритм обработки набора операндов, в которой описывается не поток инструкций, а *поток правил*, порождающих последовательность шагов обработки операндов.

Нетрадиционность позволила обойти ограничения, свойственные фон-неймановским компьютерам. Но, главное, она обеспечила свободу в выборе решений и, как следствие, обеспечила высокую степень их новизны. Все принимавшиеся решения проверялись на соответствие главной цели проекта – существенному повышению эффективности ЦОС за счёт использования положительных свойств рекуррентно-динамического подхода к организации ВП.

Из-за новой парадигмы управления вычислительным процессом рекуррентно-динамическая архитектура РВП, как уже отмечалось в [1], *не имеет аналогов* в современной коммерческой компьютерной технике.

4.1.3. Возможности развития архитектуры

В зависимости от размерности данных (единица измерения – x*D*, от англ. *dimension*), на которые ориентирована РДА, возможны следующие её подвиды**:** *скалярная* (РДА-0D), *векторная* (РДА-1D), *матричная* (РДА-2D) и *универсальная* (РДА-3D). В универсальной (тензорной) РДА в перспективе предполагается реализовать принцип *иерархической рекуррентной вложенности с охранением подобия* (принцип «русской матрёшки»). Только в этом случае могут быть обеспечены у базовой структуры ЕНК (РДА-3D) свойства *градации* (эволюции) и деградации (инволюции), которые необходимы ей и её подструктурам для обеспечения свойств *самоподобия* (*фрактальности*) [3] и *гарантоспособности* [4] у моделей ЕНК.

5. ПРИНЦИПИАЛЬНАЯ БАЗА АРХИТЕТУРЫ РВП

5.1. Ключевые принципы рекуррентно-динамической парадигмы

Свобода в выборе решений позволила отказаться от канонизировавшейся фон-неймановской концепции проектирования компьютеров, открытой для восприятия последовательных статических процессов и плохо воспринимающей динамичные процессы (как

последовательные, так и параллельные). Предлагаемая парадигма открыта для всех видов процессов**:** динамических, статических и смешанных. Она порождает систему сбалансированных принципов, позволяющих улучшить показатели качества компьютерных архитектур. Все они относятся, прежде всего, к *вычислительному процессу и его составляющим*, но применимы также к другим сторонам функционирования любых структурных объектов.

Главные (ключевые) принципы, сопутствующие новой *парадигме рекуррентно-динамических вычислений* и, соответственно, всем архитектурам на её основе**:**

(1) *динамический характер вычислительных процессов*;

(2) *рекуррентность в представлении алгоритмов, управляющих структур и внутреннего языка*;

(3) *повышение статуса данных до уровня активных* (самодостаточных) *единиц*;

(4) *автономность* (независимость и самоуправляемость) *процессов и данных*;

(5*)* *самосинхронизация процессов и циклов взаимодействия* (событийная природа циклов взаимодействия внутри и любая - при взаимодействии с внешней средой);

(6) *иерархическая вложенность* (упорядоченность по вертикали) *и холоархичность* (равенство по горизонтали)***;***

(7) *естественный* (многопоточный) *параллелизм* (теоретически неограниченное число вычислительных потоков).

Все другие, частные, принципы, являющиеся производными от этих ключевых принципов, в данной работе не рассматриваются.

Перечисленные принципы считаются *ключевыми* потому, что они определяют ход (динамику) вычислительного процесса в целом и, следовательно, влияют на все уровни организации вычислительных устройств (в том числе и РВП), обеспечивая им обобщающий принцип – *целостности.* Все они имеют смысл только в единстве, хотя ниже раскрываются как автономные единицы.

5.2. Краткая расшифровка ключевых принципов

Первый принцип (*динамичности процессов*) определяет метод решения задачи, требующий представить её алгоритм в виде *динамически разворачивающегося* графа (реально не существующего), но проявляющегося в процессе исполнения (развёртки) в виде *графовой траектории* [1, 5].

Второй принцип (*рекуррентности*) выделяет из множества объектов и процессов только такие, которые характеризуются *рекуррентной вложенностью*. Без интеграции второго принципа с первым невозможна реализация графодинамического подхода к решению задач. Применение *рекуррентной графодинамики* для представления алгоритмов задач означает существенное сокращение объёмов памяти под управляющие структуры и программы, имеющие тенденцию к быстрому усложнению в вычислительных и управляющих системах. Способ решения этой проблемы, используемый в РВП, - это рекуррентное представление граф-алгоритма и рекуррентная «свёртка-развёртка» его в процессе исполнения. Рекуррентное представление алгоритмов и процессов влечёт за собой и рекуррентный язык их описания.

Рекуррентность – ведущий принцип архитектуры РВП, поскольку она базируется на следующих решениях:

- *рекуррентное представление алгоритмов* (обеспечивается компилятором);

- *рекуррентная организация управления развитием* (развёрткой) *вычислительного процесса на операционном уровне* (за счёт саморазвёртки функциональных полей, сопровождающих данные);

- *рекуррентный внутренний машинный язык*.

Третий принцип (*повышение статуса пассивных данных до уровня самодостаточных*) отдаёт предпочтение *активным* (самодостаточным) объектам, способным самостоятельно продвигать вычислительный процесс от его начала к концу. Активный объект

способен вырабатывать собственную последовательность управления, *пассивный* – таким свойством не обладает. Вычислительный процесс в РВП строится по принципу *инициативы от данных* (самодостаточных операндов), в отличие от фон-неймановских компьютеров, где агентами активизации ВП являются инструкции. Как это управление реализуется на операционном уровне РВП, описано в [2].

Принцип повышения статуса данных подразумевает существование только одного потока – потока элементов самодостаточных данных. В момент активизации каждый ЭСД становится *саморазворачивающейся* единицей, способной самостоятельно определять пути своего перемещения и способы преобразования на каждом шаге перемещения.

Четвёртый принцип (*автономности* или *самоуправляемости*) обязывает наделять принципиально важные функциональные составляющие, участвующие в реализации ВП (структуры, объекты и процессы), свойством «*самости*» - способностью существовать, перемещаться и совершать действия за счёт собственных ресурсов. *Самоуправляемость* есть способность объекта изменять своё состояние, структуру и режимы функционирования под влиянием *собственных* управляющих воздействий. Управляемый объект способен выполнять то же самое, но под влиянием *внешних* управляющих воздействий, если это в него заранее заложено.

Несколько слов о принципе *автономности*, который есть не что иное, как принцип «*самости*», поскольку распространяется не только на типы данных, обладающие *самодостаточностью ресурсов* для *самоисполнения*, но и на физический уровень архитектуры, фундаментом которого являются интегральные схемы (ИС), получившие название *самосинхронных* [6,7]. Самосинхронные ИС удовлетворяют этому принципу такими своими свойствами, как *самопроверяемость, самодиагностируемость и самореализуемость циклов взаимодействия*. Очевидно, что принципу «*самости*» отвечают все свойства и функции объектов, начинающиеся с приставки «*само*…».

Без принципа *самодостаточности данных* невозможно достичь *самоисполняемости* алгоритмов на всех шагах их развёртки. Он – главный из всего набора принципов с приставкой «само…», поскольку интегрирует в себе функции, обеспечивающие саморазвитие процесса вычислений, саморазвёртку функциональных (теговых) полей, самосинхронное взаимодействие, самопродвижение и самообработку операндов, саморазборку и самосборку структуры вычислительного процесса на отдельных шагах обработки, самосборку наборов данных.

Пятый принцип (*о событийной природе взаимодействий*) требует использования естественных (не принудительных!) форм синхронизации взаимодействий внутри функционально автономных структур, объектов и процессов. Синхронизация посредством событий, происходящих внутри (а не вне) взаимодействующих структур, объектов и процессов, естественна для них. Событийная синхронизация – есть *самосинхронизация*. К виду синхронизации при выполнении внешних циклов взаимодействия подобное требование не всегда может предъявляться, поскольку зависит от окружения, определяющего тип связи, которая может быть одного или нескольких видов - *синхронной*, *асинхронной*, *квазисамосинхронной* либо *строго самосинхронной*.

Принцип самосихронного взаимодействия в РВП базируется на свойстве архитектур потока данных инициировать продвижение ВП по готовности операндов для дальнейшей обработки. Естественность этой схемы координации событий в том, что она не требует привлечения внешнего управления (принудительной синхронизации) для организации развития процесса. Применение в РВП принципа событийного (самосихронного) взаимодействия позволит сделать естественными развитие ВП и аппаратный параллелизм системы, а именно**:**

- *саморазвитие вычислительного процесса без внешних принуждающих воздействий*;

- *параллельность развития вычислительного процесса как множества одновременных и независимых событий*;

- *апериодическую асинхронность взаимодействий объектов ВП без ограничения длительности событий*;

- *самосогласованность взаимодействий объектов на локальных уровнях* (без привязки к какой-либо принудительной тактовой системе).

Самосинхронизация принадлежит к числу универсальных (изобретённых самой природой) принципов взаимодействия. Именно этот принцип предоставляет разработчику естественный способ распределённого управления параллельными процессами любого вида, как статическими, так и динамическими.

Шестой принцип (*иерархии - холоархии*) констатирует факт, что одна и та же совокупность элементов может рассматриваться как самостоятельная система и как часть другой, более крупной системы (надсистемы), в которую она входит. В свою очередь, эта же совокупность элементов может рассматриваться как большая система по отношению к частям, входящим в неё (подсистемам). Принцип, таким образом, означает**:** *системы вложены друг в друга и иерархичны по значимости и управляемости*. Всё сказанное распространяется и на процессы. Иерархическая организация позволяет достичь хорошей управляемости (извне) и самоуправляемости (изнутри) уровней. При этом элементы в иерархии упорядочены следующим образом**:** по уровням – *субординацией* (по вертикали), внутри уровней – *координацией* (по горизонтали). Специфицируемая архитектура имеет иерархическую организацию с числом вложенных уровней, равным четырём. Принцип холоархии утверждает возможность существования на каждом уровне произвольного числа одновременно обрабатываемых объектов равной значимости (например, в РВП вектор состоит из линейного набора самодостаточных слов, которые на операционном уровне обрабатывается параллельно, каждое слово ЭСД в своём слайсе).

Естественность иерархии архитектуры РВП следует также из естественной иерархии выполняемых процедур (функций) и связанной с ними иерархии структур данных. В этом плане операнд в РВП рассматривается как элемент саморазворачивающихся данных нижнего

(нулевого) уровня архитектуры, называемого *операционным уровнем*. Особенность этого уровня в том, что только с ним связана эволюция (от лат. *evolutio* - *развёртка*) *содержательных* частей ЭСД. Другими словами, возникновение и любые изменения содержательных частей возможны только у операндов и только на операционном уровне. Эволюция функциональных частей, то есть их рекуррентная развёртка, осуществляется на всех уровнях архитектуры (развёртке подвержены структуры данных всех рангов).

Седьмой принцип (*естественного параллелизма*) задаёт процесс (схему) преобразования исходного (последовательного) алгоритма в параллельный алгоритм. В этом процессе выявление частей, способных выполняться одновременно, осуществляется в ходе исполнения алгоритма, причём стратегия дальнейшего развития процесса – переход к следующему шагу – уточняется на каждом текущем шаге (в силу того, что на каждом текущем шаге рекуррентной развёртки формируются также данные для следующего шага). Этот принцип провозглашает независимость пространственного параллелизма от архитектурного. Параллелизм архитектуры здесь понимается в традиционном смысле [8]. В отношении параллелизма специфицируемая архитектура отличается универсальностью. Она способна воспринять все известные виды параллелизма**:** классические (SIMD- и MIMD-типов), пространственный (многопроцессорный) и временной (конвейерный, MISD-типа). Эта универсальность есть следствие объединения потоков инструкций (функций) и данных (аргументов) в единый поток ЭСД, после чего параметр двойной кратности потоков, в известном смысле, перестаёт быть классификатором параллелизма.

Хотя архитектура РВП характеризуется как параллельная (и масштабируемая), её параллелизм не является, в отличие от иерархии, обязательным свойством. Параллелизм в данном случае следует понимать как отсутствие ограничений на увеличение числа каналов параллельной обработки (масштабирование), поскольку каждый шаг обработки может осуществляться над множеством ЭСД, составляющих

текущий временной срез процесса**:** одновременно может селектироваться в пары и обрабатываться множество пар аргументов (операндов). Исходные принципы организации процессов порождают *динамический параллелизм*, вытекающий из структуры процесса (алгоритма). В РВП этот вид параллелизма извлекается из алгоритма на стадии компиляции его в код, называемый *начальным условием развёртки* (НУР) и записываемый в соответствующее место функциональной части ЭСД.

6. АЛГОРИТМИЗАЦИЯ ЗАДАЧ В СВЕТЕ РЕКУРРЕНТНО-ДИНАМИЧЕСКОГО ПОДХОДА

6.1. Новый подход к представлению алгоритмов задач

Выше уже говорилось о том, что для решения любой задачи, связанной с обработкой сигналов, необходимо её математический алгоритм (формулу или набор формул) представить в виде параллельного графа, свернуть его рекуррентным образом в «код», который и будет начальным условием развёртки РДГ (когда это потребуется). На самом же деле последовательность подготовки задачи к решению её на РВП достаточно трудоёмка и требует соответствующей квалификации. Правильнее было бы определить её не как последовательность, а как методику подготовки задач ЦОС к решению на РВП. Её этапами (при ручной подготовке) могли бы быть**:**

1. *Математическая постановка задачи* (перевод словесно поставленной задачи на математический язык).

2. *Разработка аналитической модели* (набора формул) *решения задачи* (либо выбор подходящей модели из числа имеющихся).

3. *Составление граф-алгоритма решения задачи* - графа зависимости операндов (ГЗО)*, естественным образом вытекающего из формул (чаще всего это последовательно-параллельный граф).

4. *Выполнение операции упорядочения вычислений по естественному графу* (с упрощением графа, насколько это возможно).

5. *Преобразование естественного графа в параллельный* (максимальное извлечение параллелизма из алгоритма с привлечением соответствующих методов преобразования [9]).

6. *Составление локального или локально-рекурсивного* ГЗО для параллельного алгоритма (*обеспечение замыкания данных внутри капсулы-задачи*) и сопутствующего ему *предварительного графа развёртки* (*ГР*)*.

7. *Отображение параллельного ГЗО алгоритма на аппаратную структуру вычислительной среды РВП* (векторную, матричную).

8. *Представление параллельного ГЗО-ГР в виде целочисленной рекуррентной последовательности* (сворачивание графа в цифровой код, именуемый «*начальным условием развёртки*» алгоритма задачи).

9. *Упаковка кода* НУР *конкретной задачи* в слово ЭСД.

* ГЗО обеспечивает графическое представление вычислительного алгоритма, показывая взаимосвязи по данным, а ГР - отображает процесс пошаговой развёртки программы, управляющей выполнением ГЗО.

Естественно, чтобы рекуррентно закодировать алгоритм (преобразовать его до размеров кода НУР, размещаемого вместе со стартовым операндом в пределах одного машинного слова ЭСД), нужно уметь его «сворачивать». Рекуррентная свёртка алгоритмов – процесс трудоёмкий, если его выполнять вручную. Но если такая процедура автоматизирована (то есть механизм рекуррентной свертки является частью компилятора, преобразующего коды внешнего языка описания графодинамических задач в машинные коды), то выполнение процесса кодирования резко упрощается. Он становится доступным каждому программисту. Практика показала - для автоматизации процесса свертки достаточно возможностей персональных ЭВМ.

6.2. Новый подход к исполнению алгоритмов задач

При динамическом подходе к решению задачи, как отмечалось выше, граф её алгоритма в компьютере в явном виде не существует (в развёрнутом виде в памяти его нет). Он динамически возникает и исчезает в процессе реализации алгоритма по шагам. Такое поведение

графа (пошаговая саморазвёртка) обеспечивается его заблаговременной свёрткой (по методике, описанной выше) в «точку» за определённое число шагов. «Точка» на самом деле является машинным словом, в котором записан код, полученный на последнем шаге свёртки и являющийся ***н****ачальным* ***у****словием само****р****азвёртки* алгоритма («код НУР»). Инициаторами развёртки на каждом шаге являются *данные* (операнды), статус которых повышен до роли *активных единиц*, наделённых свойством *самодостаточности*. Фактически в каждый конкретный момент времени РДГ отображает одномоментное состояние процесса реализации алгоритма, а развитие самого процесса – динамика – связано не с движением по графу (что имеет место при статическом подходе к реализации алгоритма), а с *изменением самого графа*. Следовательно, объектом реализации в динамически представленной задаче, то есть переменной величиной, изменение которой во времени описывает динамический процесс, служит *граф в целом*. Последовательность одномоментных состояний образует *графовую траекторию* развития процесса. В аппаратуре РВП (ЕНК) этот процесс соотносится с *вычислительным процессом*.

Положительная составляющая рекуррентно-динамического подхода заключается в том, что он позволяет эффективно решать задачи, в которых во времени могут происходить динамические изменения их структуры. Примеры такого рода задач приведены в [1].

Следует отметить, что многие такие структуры адекватно отображаются тем или иным видом графа, но чаще всего это - ориентированные виды графов (графы-«деревья» или графы типа «лес»). Рекуррентно-динамический подход универсален, ему по плечу все статические и статико-динамические процессы и алгоритмы. Однако во всех случаях необходимо уметь представлять реальные процессы в виде последовательности $\boldsymbol{x}(\boldsymbol{t})$ одномоментных графов и затем рекуррентно сворачивать её в начальный (запускающий) граф $\boldsymbol{x}(\boldsymbol{0})$. Запуск графа $\boldsymbol{x}(\boldsymbol{0})$ обеспечивает реализацию графовой траектории

(последовательности графов $\boldsymbol{x}(\boldsymbol{t})$), то есть реализацию алгоритма, а следовательно, и задачи.

6. 3. Динамическая самосборка наборов данных и капсул

Недостаточно наделить капсулы, извлекаемые из информационной базы (ИБ) компьютера, свойством самоисполнения. Необходим механизм, который обеспечивал бы и обратный процесс – самосборку новых капсул из потока новых слов ЭСД, возникающих на каждом шаге обработки пар операндов. Дело в том, что разборка любой капсулы порождает поток слов ЭСД, в каждом из которых содержится свой НУР, который, в свою очередь, запускает многошаговую развёртку конкретного алгоритма задачи. По завершении работы каждого алгоритма образуются результаты, новый операнд или множество операндов, одним словом «*наборы данных*» (последние могут быть как пассивного, так и активного типа). С выхода операционного устройства данные, по мере их появления, должны отправляться далее к месту их приёма - либо на объект управления (после преобразования в сигналы посредством ЦАП), либо снова в ИБ компьютера. Реализация процедуры самосборки НД и преобразования их в самодостаточные данные, капсулы (задачи) и задания (наборы задач) обеспечиваются в РВП функциями процедурного и диспетчерского уровней его архитектуры.

Заметим, что если процедуры разборки и сборки полностью разделены (реализуются в полностью независимых аппаратных каналах), то работа РВП может быть полностью конвейеризована.

Более подробно состав уровней архитектуры РВП, их функции и структурная организация, а также типы машинных данных и их роль в организации ВП освещены в [10]. Заметим только, что РДА полностью полагается на иерархическую структуру данных, упорядоченных по вертикали и горизонтали. Таблица, приводимая ниже, даёт общее представление об используемых типах данных и их упорядоченности по уровням. Каждому уровню соответствуют и свои языки

программирования, в совокупности образующие единый язык программирования капсул РВП.

Для ознакомления с другими особенностями организации РДП, РДА и РВП на её основе рекомендуем обратиться к [1, 2, 10, 11].

7. Выводы

Перечисленные выше особенности архитектуры РВП, проектируемой на основе рекуррентно-динамической парадигмы вычислений, были формулированы на основе результатов исследований, частично изложенных в упомянутых выше работах. Полученные научные и практические результаты, а также опыт и интуиция подсказывают, что рекуррентно-динамическая парадигма вычислений расширяет диапазон возможностей конструкторов вычислительных устройств ЦОС, позволяя им**:**

* *создавать многопоточные событийно-ориентированные* (самосинхронные) *системы, поддерживающие массовый параллелизм при меньших ресурсах и накладных расходах***;**

* *уменьшать* (причём существенно) *объем памяти СБИС-систем за счет компрессионной* (рекуррентной) *формы представления алгоритмов* (программ)**;**

* *сократить потери времени на пересылку, загрузку и хранение «программ» и данных за счет их компрессии*;

* *повысить защищенность информации и алгоритмов программ* (в том числе от вирусов) *за счет использования рекуррентного принципа кодирования*;

* *избавиться от проблемы связности программ и данных, замыкая их внутри капсул*.

Перечисленные возможности чрезвычайно важны для любых классов компьютерных систем, но без них нельзя создать сегодня компьютерные системы управления ответственными объектами и процессами, отвечающие требованию гарантоспособности. Согласно [4], *гарантоспособность* – обобщающее свойство компьютерной системы, характеризующее возможность её применения в системах управления ответственными процессами и объектами.

Предварительное исследование рекуррентно-динамической архитектуры РВП на основе РДП для ЦОС показало: предлагаемая архитектура может быть охарактеризована как *de novo* и стоит того, чтобы её реализовать, особенно на фоне имеющего место кризиса идей в области архитектур параллельных компьютеров.

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

3. *Федер Е.* Фракталы. – М.: Мир, 1991. – 263 с.

4. *Avizienis A., Laprie J.C.* Dependable Computing: From Concepts to Design Diversity. // Proc. IEEE. – 1986. – Vol. 74, № 5. - P. 629-638.

5. Исследования в области архитектуры персональных ЭВМ и основы структурной динамики. / Яковлев Ю.С., Махиборода А.В. – Киев, 1987. – 31 с. (Препринт АН УССР, Ин-т кибернетики им В.М. Глушкова; 87-35).

6. Автоматное управление асинхронными процессами в ЭВМ и дискретных системах. / Под ред. В.И. Варшавского. – М.: Наука, 1986. –398 с.

7. *Филин А.В., Степченков Ю.А*. Схемотехника интегральной элементной базы естественно-надёжных компьютеров. // Системы и средства информатики. Вып.7. –М.: Наука, 1995. – с. 222 – 239.

8. *Flynn M.J.* Some Computer Organization and Their Effectiveness. IEEE Transaction on Computers, C-21, 9 (1972), pp. 948 – 960.

9. *Кун С.* Матричные процессоры на СБИС. Пер. с англ. – М.: Мир, 1991. – 672 с.

10. *Филин А.В*. Особенности организации обработки сигналов на процессоре с рекуррентно-динамической парадигмой вычислений. // Наст. сб.

11. *Рождественский Ю.В., Рождественскене А. В.* Новый подход к анализу параллельных процессов в самосинхронных схемах.// Наст. сб.