Федеральное агентство по образованию
Государственное образовательное учреждение
высшего профессионального образования
Владимирский государственный университет

А.К. ФИЛИППОВ

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОЕКТИРОВАНИЯ ДИНАМИЧЕСКИ РЕКОНФИГУРИРУЕМЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ

Учебное пособие

Владимир 2010

УДК 004.031.6
ББК 32.85+32.973.2
Ф53

Рецензенты:
Доктор технических наук, профессор,
проректор по информатизации Российского нового университета
С. В. Дворянкин

Доктор технических наук, профессор кафедры радиотехники
и радиосистем Владимирского государственного университета
А. К. Бернюков

Печатается по решению редакционного совета
Владимирского государственного университета

Филиппов, А. К.

Ф53 Теоретические основы проектирования динамически реконфигурируемых систем обработки информации : учеб. пособие / А. К. Филиппов; Владим. гос. ун-т. – Владимир : Изд-во Владим. гос. ун-та, 2009. − 119 с.

ISBN 978-5-9984-0008-7

Посвящено одному из наиболее перспективных направлений в современной вычислительной технике – динамически реконфигурируемым системам обработки информации. Рассмотрены теоретические основы функционирования подобных систем, а также их архитектурные особенности. Представлены описания различных областей применения динамически реконфигурируемых систем обработки информации.

Предназначено для студентов 4-го курса очной формы обучения специальностей 210201 − проектирование и технология радиоэлектронных средств; 210202 − проектирование и технология электронно-вычислительных средств, а также для магистрантов 1, 2-го курсов, обучающихся по направлению 210200 – проектирование и технология электронных средств. Может быть полезно при изучении следующих дисциплин: «Проектирование центральных и периферийных устройств электронно-вычислительных средств», «Интегральные устройства радиоэлектроники», «Проектирование интегральных микросхем и микропроцессоров», «Теория динамически реконфигурируемых систем», «Системы компрессии и декомпрессии данных».

Табл. 5. Ил. 37. Библиогр.: 74 назв.

УДК 004.031.6
ББК 32.85+32.973.2

ISBN 978-5-9984-0008-7

Список принятых сокращений

АВУ – адаптивное вычислительное устройство
АЛУ – арифметико-логическое устройство
АО – алгоритмическое обеспечение
АСД – адаптивное сжатие данных
АЦП – аналого-цифровой преобразователь
ДРВ – динамически реконфигурируемый вычислитель
ДР ПЛИС – динамически реконфигурируемая программируемая логическая интегральная схема
ДРС – динамически реконфигурируемая система обработки информации
ДРМ – динамически реконфигурируемый модуль
ЗУ – запоминающее устройство
КВУ – комплементарное вычислительное устройство
НРС – нормальное рабочее состояние
ОЗУ – оперативное запоминающее устройство
ООП – объектно-ориентированное программирование
ОП – оперативная память
ОСРВ – операционная система реального времени
ПД – память данных
ПЗУ – постоянное запоминающее устройство
ПК – память конфигураций
ПЛИС – программируемая логическая интегральная схема
ПП – память программ
ППП – память программ процессора
ПТФ – прямые тригонометрические функции
ПЦОС – процессор цифровой обработки сигналов
РСП – реконфигурируемый сопроцессор
РУМ – резервный управляющий модуль
СБИС – сверхбольшая интегральная схема
СНАС – самонастраивающаяся адаптивная система
СНР – состояние неработоспособности
СОАС – самоорганизующаяся адаптивная система
УО – устройство обработки
УАСД – устройство адаптивного сжатия данных
ЦАП – цифро-аналоговый преобразователь
ЦСП – цифровой сигнальный процессор
ЭВМ – электронно-вычислительная машина
ЭВС – электронно-вычислительное средство

CISC – Complete (Complex) Instruction Set Computer
DISC – Dynamic Instruction Set Computer
DMA – Direct Memory Access
DPMC – Dual-Port Memory Controller
DSP – Digital Signal Processor
EAB – Embedded Array Block
FIFO – First Input First Output
FPGA – Field Programmable Gate Array
FRAM – Ferromagnetic Random Access Memory
HPI – Host-Port Interface
I^2C – Inter-IC bus
ISA – Industrial Standard Architecture
JPEG – Joint Photographic Experts Group
JTAG – Joint Test Automation Group
LZW – Lempel-Ziv-Welch encoding
MAC – Multiplication and ACcumulation
MIMD – Multiply Instruction Multiply Date
MPEG – Motion Pictures Experts Group
NOP – No OPerations
PAM – Programmable Active Memories
PCI – Peripheral Component Interconnect
PE – Processing Element
RAM – Random Access Memory
RISC – Reduced Instruction Set Computer
RLE – Run-Length Encoding
RTOS – Real-Time Operating System
RTCP – Real-Time Control Protocol
RTP – Real-time Transport Protocol
RTSP – Real-Time Streaming Protocol
SCU – Standby Control Unit
SDRAM – Synchronous Dynamic Random Access Memory
SISD – Single Instruction Single Date
SIMD – Single Instruction Multiply Date
SPI – Serial Peripheral Interface
SPIHT – Set Partitioning in Hierarchical Trees
SRAM – Static Random Access Memory
VHDL – Very high speed integrated circuits Hardware Description Language

ВВЕДЕНИЕ

Появление таких устройств, как динамически реконфигурируемые системы обработки информации (ДРС), было обусловлено самой логикой развития вычислительной техники. Если провести анализ теоретических основ, на которых базируются традиционные вычислители, то становится понятно, что их архитектура, заложенная еще в 1945 году Джоном фон Нейманом, представляет собой один из многих возможных вариантов реализации принципа Чёрча-Тьюринга, описывающего то, какими свойствами должен обладать компьютер, способный решить любую алгоритмически разрешимую задачу. Следует подчеркнуть, что те архитектурные решения, которые применил Джон фон Нейман, в значительной степени были обусловлены существовавшими тогда технологическими ограничениями. В этой связи создание ДРС можно рассматривать как производную того прорыва в области технологии изготовления цифровых СБИС, который наблюдался в течение 80 − 90-х годов XX века.

С точки зрения математики, главная особенность традиционных процессоров состоит в однозначности интерпретации составных операций, которые обязательно представляются как совокупность элементарных преобразований, задаваемых математической формулой, − то есть принцип декомпозиции применяется напрямую. Ограниченные возможности подобного подхода становятся очевидными на примере процессоров с широко распространенной системой команд x86: при решении достаточно сложных задач, требующих режима реального времени (в частности, задач обработки видео- и аудиоданных), возникла необходимость в расширении набора команд. С этой целью были разработаны такие специальные проблемно-ориентированные наборы операций, как MMX, SSE, SSE2, SSE3 (у процессоров фирмы Intel) и 3DNow, 3DNow+ (у процессоров фирмы AMD). Функционально те же самые операции могут быть реализованы с помощью основного набора команд, но время выполнения будет слишком велико.

Таким образом, основная цель этих расширений – сокращение времени выполнения составных операций за счет изменения как математических методов расчета, так и вариантов их реализации. Подход, заключающийся в создании проблемно-ориентированных расширений системы команд, обладает двумя существенными недостатками:

1) увеличение аппаратных затрат, а, следовательно, себестоимости процессора;
2) крайне ограниченная область применения подобных расширений.

Другого рода ограничения связаны с реализацией алгоритмов, обладающих различными видами параллелизма и предполагающих использование различных структур операционного автомата. Как известно, даже современные процессоры, как правило, ориентированы только на один вид параллелизма и тип структурной организации операционного автомата, что также негативно сказывается на реальной производительности традиционных вычислительных устройств. Все вышеуказанные недостатки в различной степени могут быть устранены за счет применения ДРС, которые фактически являются расширением процессоров традиционной архитектуры.

Следует отметить, что в настоящей работе в виду ограниченности объема не рассматриваются подробно вопросы выбора элементной базы для ДРС. Это связано, прежде всего, с наличием широкого выбора литературы по данной тематике как отечественных [39, 40], так и зарубежных [73, 74] авторов. Кроме того, наиболее распространенный вариант построения ДРС – на базе процессоров и динамически реконфигурируемых программируемых интегральных схем – не является единственным, что открывает перспективы применения тех же принципов в другом конструктивно-технологическом исполнении (например, на основе оптоэлектронных устройств [57]).

Предполагается, что студенты, приступающие к изучению дисциплины, должны быть знакомы с теоретическими основами проектирования электронно-вычислительных средств и информатики, а также с современной электронной элементной базой (в частности, с динамически реконфигурируемыми программируемыми логическими интегральными схемами, микропроцессорами, различными типами энергозависимой и энергонезависимой памяти).

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ФУНКЦИОНИРОВАНИЯ ДИНАМИЧЕСКИ РЕКОНФИГУРИРУЕМЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ

К настоящему времени в области вычислительной техники наметились качественные изменения: с одной стороны, бурно внедряются новые технологические процессы, в основе которых лежат передовые методы микро- и наноэлектроники, а с другой, наблюдается стагнация в развитии архитектурных решений для вычислительных систем, что позволяет говорить о кризисе традиционных процессорных устройств [50]. При этом вряд ли стоит разделять оптимизм некоторых исследователей, которые полагают, что использование квантовых свойств в расчетах и, соответственно, переход к так называемым квантовым компьютерам решат большинство или даже все насущные проблемы вычислительной техники, – слишком мало количество приложений, в которых квантовый компьютер дает выигрыш в производительности перед традиционными вычислителями [21]. В этой связи практически актуальной и научно важной проблемой становится разработка новых архитектур, которые могли бы быть реализованы как на существующей, так и на перспективной элементной базе [50]. Одним из подходов к решению данной проблемы является создание новых вычислительных устройств на базе динамически реконфигурируемых систем [50].

1.1. Определение динамически реконфигурируемой системы обработки информации

Прежде, чем приступать к подробному рассмотрению особенностей и преимуществ динамически реконфигурируемых систем, которыми они обладают по сравнению с традиционными вычислителями, необходимо дать их точное определение. ***Динамически реконфигурируемые системы*** (dynamically reconfigurable systems, adaptive computing systems) – это класс вычислителей, способных менять свою внутреннюю логическую структуру непосредственно в процессе

функционирования [50] за время, значительно меньшее времени выполнения вычислительных задач, между которыми происходила смена структуры [47, 58]. Следовательно, предельно допустимое для ДРС значение времени конфигурирования $T_{\text{CONF}}^{\text{TH}}$ в общем случае может изменяться с течением времени: $T_{\text{CONF}}^{\text{TH}} = f(t)$. Таким образом, система является динамически реконфигурируемой, если справедлива следующая система неравенств:

$$\begin{cases} T_{\text{CONF}i}^{\text{TH}} << T_{\text{TSK}i}, \text{ где } i = 1, \\ T_{\text{CONF}i}^{\text{TH}} << \min\{T_{\text{TSK}i-1}, T_{\text{TSK}i}\}, \text{ где } i = \overline{2,n} \end{cases}, \tag{1.1}$$

где $T_{\text{TSK}i}$ – время выполнения i-й вычислительной задачи. Следовательно, требования к значению времени реконфигурирования ДРС могут меняться во время работы устройства. Если система неравенств (1.1) не выполняется, то нельзя говорить, что система обладает свойством динамической реконфигурации. Таким образом, является ли система динамически реконфигурируемой или нет, в равной степени зависит и от характеристик аппаратной платформы, и от решаемых на ней задач [47, 58].

Любая ДРС состоит как минимум из двух базовых элементов (рис. 1.1) [33]:

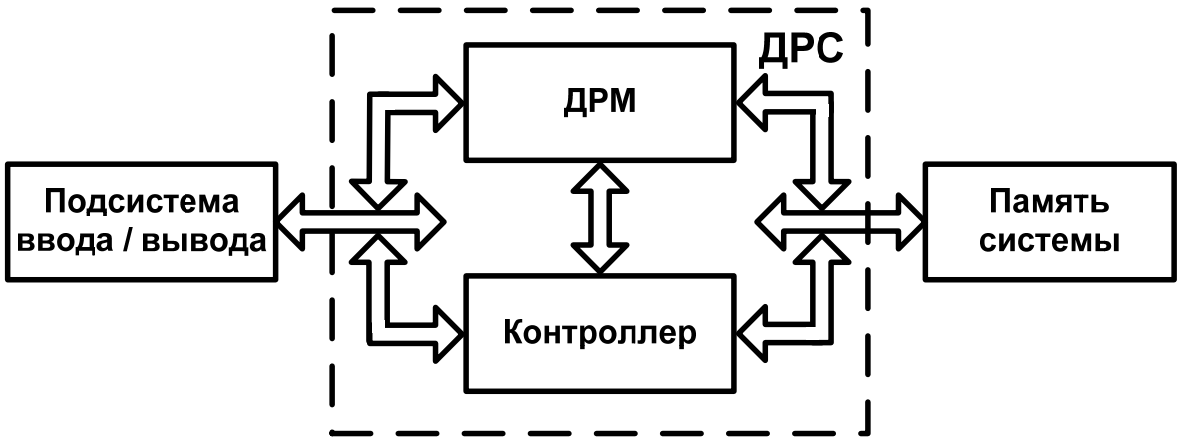


Рис. 1.1. Обобщенная структурная схема ДРС

- ***динамически реконфигурируемого модуля (вычислителя),*** который предназначен для реализации вычислительных операций и основной элементной базой которого на сегодняшний день являются ДР ПЛИС;

- ***управляющего модуля (контроллера)***, в котором реализуются основные операции управления и специальная операция загрузки конфигурационного файла в динамически реконфигурируемый модуль из произвольного места в памяти. Таким образом, контроллер фактически представляет собой процессор, функциональный базис которого может быть неполным (могут отсутствовать все вычислительные операции, кроме тех, которые необходимы при работе с памятью).

Безусловно, что динамически реконфигурируемый модуль – это тот самый элемент, который отличает ДРС от традиционных процессорных устройств, хотя реализация такого фундаментального свойства вычислителей, как ***динамическая реконфигурация***, возможна только при наличии обоих элементов.

Говоря о приложениях ДРС, можно утверждать, что на ее базе целесообразна реализация только таких задач, которые требуют ***структурной адаптации*** (то есть изменения структуры) вычислительного модуля. К ним среди прочего относятся следующие:

− *адаптивная обработка данных*: в частности, адаптивные алгоритмы цифровой обработки сигналов и другой информации; адаптивные системы управления (прежде всего, самоорганизующиеся системы);

− *сокращение аппаратных затрат на реализацию алгоритмов за счет последовательного использования одного и того же аппаратного обеспечения*: например, аппаратная реализация различных сетевых протоколов транспортного уровня.

В любом случае динамически реконфигурируемые системы обработки информации – это совокупность архитектур, ориентированных на реализацию устройств с возможностью адаптации. Как известно, адаптивные системы подразделяются на две большие группы [45]:

– системы, использующие адаптацию ***без обратной связи*** (рис. 1.2);

– системы, использующие адаптацию ***с обратной связью*** (рис. 1.3).

Процесс адаптации без обратной связи состоит в следующем [41, 45]:

1) проводятся измерения характеристик входного сигнала и определяются его необходимые параметры;

2) полученная зависимость формализуется в виде алгоритма адаптации;

3) алгоритм адаптации реализуется как автономный модуль и регулирует устройство обработки в соответствии с заложенной зависимостью.

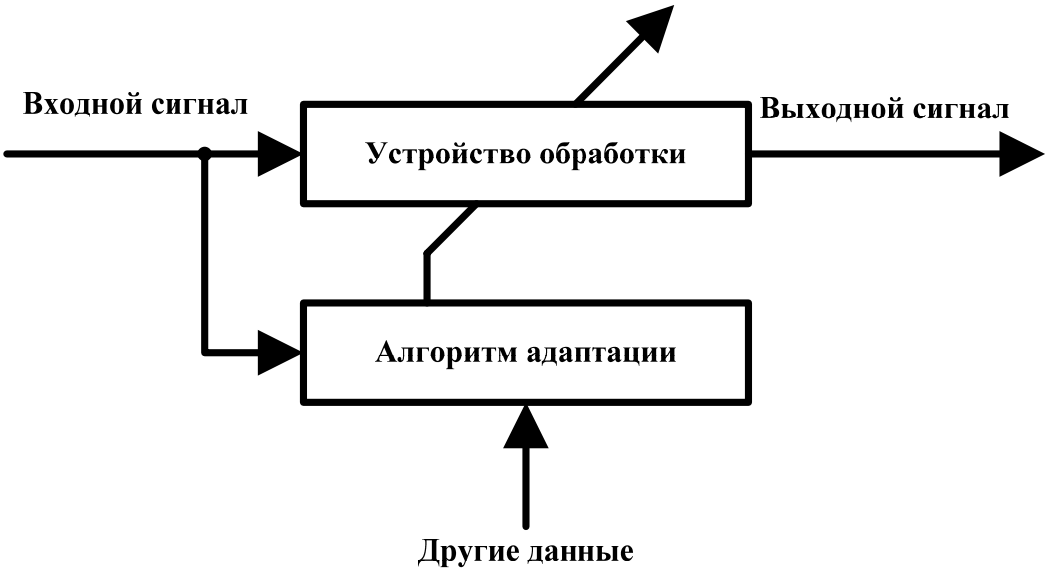


Рис. 1.2. Обобщенная схема адаптивной системы без обратной связи

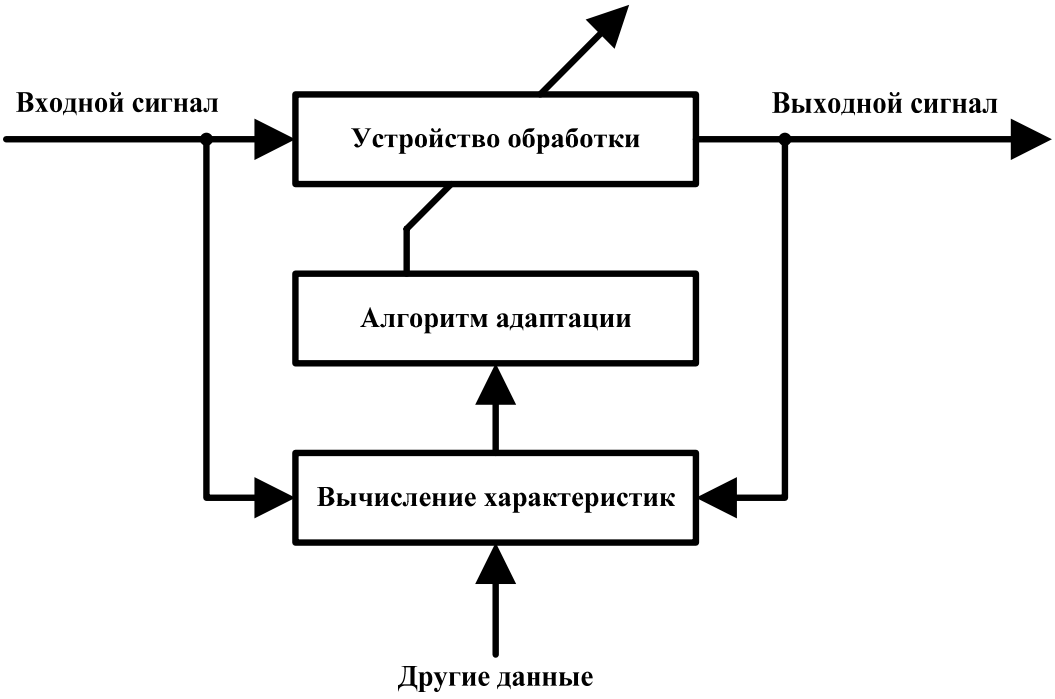


Рис. 1.3. Обобщенная схема адаптивной системы с обратной связью

При адаптации с обратной связью коррекции в алгоритм адаптации вносятся автоматически в рабочем режиме, и определяется их влияние на выходной сигнал с целью оптимизации параметров функционирования системы [41]. В литературе, как правило [25, 41], значительное внимание уделяется разработке именно алгоритма адаптации, который справедливо считается наиболее интеллектуальной частью системы. Однако при реализации алгоритмов адаптивной обработки следует учитывать взаимосвязь между алгоритмом адаптации и УО входного сигнала, поскольку от каждого из них зависит, будет ли способна система обрабатывать данные в режиме реального времени или нет. Перенося подобное представление на ДРС и сопоставляя функциональное назначение всех элементов, можно прийти к выводу, что при реализации адаптивных систем ДРМ выступает в качестве УО с перестраиваемой структурой [45].

Однако не все адаптивные системы целесообразно реализовывать на базе ДРС. Для того чтобы это наглядно показать, рассмотрим различные варианты регулирования УО, а именно то, какое влияние может оказывать алгоритм адаптации на УО адаптивной системы. Для этого приведем еще одну классификацию.

В зависимости от типа регулирующего воздействия, которое оказывает алгоритм адаптации на УО, и от структурной организации УО адаптивные системы могут быть реализованы как ***самонастраивающиеся*** и ***самоорганизующиеся***. Принцип функционирования самонастраивающихся адаптивных систем заключается в изменении некоторого набора параметров УО при сохранении его структуры в неизменном виде (на практике это соответствует, например, изменению коэффициентов цифрового фильтра). Для самоорганизующихся адаптивных систем характерно, напротив, формирование новой структуры УО при его регулировании с помощью алгоритма адаптации, что соответствует, например, замене цифрового фильтра одного типа на другой. Становится очевидным, что ДРС избыточны для реализации САС, так как в последних отсутствует необходимость в изменении внутренней структуры. В тоже время ДРС являются вполне сбалансированным решением для построения САС [45].

Вернемся к структуре динамически реконфигурируемой системы обработки информации. Если основным узлом, выполняющим функции УО, в ДРС является ДРМ, то относительно реализации алгоритма адаптации вопрос остается открытым. Априори можно утверждать, что способы его реализации могут значительно отличаться друг от друга и определяются его вычислительной сложностью. В общем случае алгоритмы адаптации могут быть реализованы программно, аппаратно и аппаратно-программно. Достоинства и недостатки каждого варианта известны и широко описаны в литературе (например, в [7, 38, 45]). Собственно говоря, выбор способа реализации алгоритма адаптации и является первой задачей разработчика электронно-вычислительного средства при создании ДРС. От адекватности этого выбора зависит качество функционирования всего устройства в целом.

Перед тем как перейти к вопросам формализации вычислительного процесса у ДРС, следует сделать три важных замечания, которые касаются структуры подобных систем. Во-первых, представление о строгом распределении функций между модулями, изложенное впервые в [33] и кратко рассмотренное выше, во многом является упрощенным. Его основное назначение состояло в том, чтобы обосновать принципиальные отличия ДРС от процессоров, базирующихся на классических принципах Джона фон Неймана. Безусловно, функции управляющего модуля и ДРМ в значительной степени оказываются «перемешанными»: многие вычислительные задачи (особенно во время конфигурирования ПЛИС) решаются контроллером, в качестве которого целесообразно применять RISC-процессор (обоснование этого будет приведено в следующей главе), а некоторые из задач управления – ДРМ.

Во-вторых, не стоит преувеличивать значение элементной базы для ДРС: такое свойство вычислителей, как динамическая реконфигурация, фундаментально и отражает общие архитектурные особенности системы, в то время как элементная база при всей ее важности определяет лишь конкретные технико-экономические характеристики изделия.

В-третьих, описанная структура ДРС ни в коем случае не накладывает ограничений на сложность системы, то есть не определяет ко-

личества процессоров, применяемых в качестве управляющего модуля, и числа ПЛИС, из которых состоит ДРМ, а также конкретных типов и характеристик взаимодействия между ними. Единственным требованием при увеличении структурной сложности ДРС является отсутствие противоречивости в управлении системой.

1.2. Математические принципы, лежащие в основе динамически реконфигурируемых систем обработки информации

Пусть система команд ДРС представляет собой множество, включающее в себя все команды ДРС, доступные в течение некоторого промежутка времени Δt_k:

$$\mathbf{A}^{\Delta t_k} = \left\{ a_i^{\Delta t_k} \right\}_{i=0}^{n(\Delta t_k)-1}, \tag{1.2}$$

где $a_i^{\Delta t_k}$ – команда ДРС под номером i, $n(\Delta t_k)$ – общее число команд в системе команд. Тогда, проводя анализ обобщенной структурной схемы ДРС, можно заметить, что $\mathbf{A}^{\Delta t_k}$ является подмножеством некоторого надмножества $\mathbf{A} = \{a_i\}_{i=0}^{N-1}$:

$$\mathbf{A}^{\Delta t_k} \subset \mathbf{A}, \tag{1.3}$$

которое включает в себя все команды ДРС a_i, реализуемые на базе ДРС в процессе её функционирования, и мощность которого равна N. $\mathbf{A}$ – это конечное множество, то есть $N \in \aleph$, что следует из конечности системы команд контроллера ДРС и конечной логической ёмкости ДРМ [49]. Предполагая постоянство системы команд контроллера ДРС, можно утверждать следующее [51]:

$$\mathbf{A} = \mathbf{A}_{\text{const}} \bigcup \mathbf{A}_{\text{adapt}}, \tag{1.4}$$

где $\mathbf{A}_{\text{const}} = \{a_{\text{const}\,i}\}_{i=0}^{M-1}$ – набор команд контроллера ДРС с мощностью M; $\mathbf{A}_{\text{adapt}} = \{a_{\text{adapt}\,i}\}_{i=0}^{N-M-1}$ – набор команд ДРМ, который адаптируется к решаемым задачам и мощность которого равна $(N - M - 1)$. Адаптивность ДРС заключается в том, что из всего множества команд ДРМ $\mathbf{A}_{\text{adapt}}$ выбирается некоторое подмножество $\mathbf{A}_{\text{adapt}}^{\Delta t_k} = \left\{ a_{\text{adapt}\,i}^{\Delta t_k} \right\}_{i=0}^{p(\Delta t_k)-1}$, которое будет действительно в течение промежутка времени Δt_k и где $p(\Delta t_k) < N - M$. Таким образом, система команд ДРС, действительная

в течение некоторого промежутка времени Δt_k, представляет собой объединение двух подмножеств [49]:

$$\mathbf{A}^{\Delta t_k} = \mathbf{A}_{\text{const}} \bigcup \mathbf{A}_{\text{adapt}}^{\Delta t_k} = \left\{ a_{\text{const } i} \right\}_{i=0}^{M-1} \bigcup \left\{ a_{\text{adapt } j}^{\Delta t_k} \right\}_{j=0}^{p(\Delta t_k)-1}. \qquad (1.5)$$

Общее число команд, доступных в течение некоторого промежутка времени Δt_k, равно $n(\Delta t_k) = p(\Delta t_k) + M$. Следует отметить, что в любой момент времени t система команд ДРС должна обладать свойством функциональной полноты. На практике свойством функциональной полноты обладает даже система команд управляющего модуля, поскольку, как было сказано выше, он строится на базе обычного RISC-процессора, возможности которого расширены за счет интеграции специальных команд [49].

Рассмотрим подробнее подмножество $\mathbf{A}_{\text{adapt}}^{\Delta t_k}$ системы команд ДРС, формируемое из надмножества $\mathbf{A}_{\text{adapt}}$:

$$\mathbf{A}_{\text{adapt}} \xrightarrow{\mathbf{F}} \mathbf{A}_{\text{adapt}}^{\Delta t_k}, \qquad (1.6)$$

где $\mathbf{F}$ – преобразование, которое заключается в поиске для некоторого вектора решаемых задач $\mathbf{T}$ бинарного маскирующего вектора $\mathbf{W}$ с длиной, равной мощности множества $\mathbf{A}_{\text{adapt}}$: $\mathbf{W} = \left\| w_j \right\|_{\overline{j=0,N-M-1}}$, где $w_j \in \{0,1\}$. Номера ненулевых элементов вектора соответствуют номерам команд ДРМ из множества $\mathbf{A}_{\text{adapt}}$, выбранным для использования в течение промежутка времени Δt_k. В общем виде преобразование $\mathbf{F}$ является трудно формализуемым, поскольку его вид определяется множеством факторов (например, количеством доступных команд ДРМ, реализованных в виде конфигурационных файлов для ДР ПЛИС, их функциональным назначением и техническими характеристиками, системой команд управляющего модуля). На практике число реально доступных команд ДРМ $\mathbf{A}_{\text{adapt}}^{\text{real}}$ значительно меньше числа теоретически доступных команд $\mathbf{A}_{\text{adapt}}^{\text{theor}}$: $\mathbf{A}_{\text{adapt}}^{\text{real}} << \mathbf{A}_{\text{adapt}}^{\text{theor}}$, что позволяет существенно упростить решаемую задачу, делая её близкой к построению обычного компилятора. Однако при разработке компилятора для ДРС следует учесть возможность расширения набора доступных команд ДРМ. Одна из основных функций компилятора для

ДРС сводится к генерации вектора, длина которого равна мощности множества $\mathbf{A}_{\text{adapt}}^{\text{real}}$:

$$\mathbf{W} = \mathbf{f}\left(\mathbf{T}, t, \mathbf{A}_{\text{adapt}}^{\text{real}}, \mathbf{A}_{\text{const}}\right), \tag{1.7}$$

где $\mathbf{T}$ – вектор решаемых задач; t – дискретное время; ***в данном контексте:*** $\mathbf{A}_{\text{adapt}}^{\text{real}}$ – вектор, в котором формализованы технические характеристики реально доступных команд ДРМ; $\mathbf{A}_{\text{const}}$ – вектор, в котором формализованы технические характеристики управляющего модуля. Как видно из формулы (1.7), векторная функция **f** является функцией дискретного времени, у которого минимальный шаг равен величине $1/f_{\text{ctrl}}$, где f_{ctrl} – тактовая частота контроллера ДРС [49]. В качестве формализованного представления вектора решаемых задач $\mathbf{T}$ может применяться входное описание проекта на языке программирования высокого уровня (например, C/C++ или Java) или/и языке описания аппаратных средств (например, VHDL или Verilog). Подводя промежуточный итог проведенному выше анализу, можно утверждать, что ДРС можно рассматривать как совокупность процессоров, обладающих идентичными интерфейсами взаимодействия с внешними устройствами, но различными системами команд и поочередно сменяющих друг друга в процессе обработки информации. Таким образом, классические процессоры являются лишь частным случаем ДРС, имеющим существенные ограничения в функциональных возможностях.

Вопросы и задания для самоподготовки

1. Дайте определение динамически реконфигурируемым системам обработки информации. Как можно в формальном виде описать особенности ДРС? Чем определяется принадлежность вычислительного устройства к классу динамически реконфигурируемых систем обработки информации? Обоснуйте Ваше мнение, опираясь на определение ДРС.

2. Нарисуйте и опишите обобщенную структурную схему ДРС. Какие базовые элементы обязательно входят в состав ДРС? Объясни-

те их функциональное назначение. На какой элементной базе они могут быть построены?

3. Приведите примеры приложений, в которых целесообразно использование ДРС. Объясните, почему.

4. Чем отличаются самонастраивающиеся и самоорганизующиеся адаптивные системы? Для каких из них применяются ДРС в качестве средства реализации и почему? Каким образом реализуются устройство обработки и алгоритм адаптации?

5. Чем отличаются системы команд ДРС и традиционного процессора, базирующегося на принципах Джона фон Неймана? Чем обусловлены эти различия?

6. Опишите математически основные отличия между ДРС и классическими процессорами. Как Вы думаете, какие преимущества и недостатки ДРС следуют из этого описания?

7. Каким образом выбирается та часть системы команд ДРС, которая реализуется на базе динамически реконфигурируемого модуля?

8. Какой инструментарий может использоваться при моделировании ДРС для формального представления вектора решаемых задач? Какие инструменты на Ваш взгляд предпочтительнее? Почему?

9. Как Вы понимаете утверждение о том, что процессоры, базирующиеся на традиционных принципах фон Неймана, представляют собой лишь частный случай ДРС? Поясните Ваше мнение.

ГЛАВА 2. АРХИТЕКТУРЫ ДИНАМИЧЕСКИ РЕКОНФИГУРИРУЕМЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ

Сейчас науке известно достаточно большое число архитектур ДРС [10 − 13, 31, 32, 43, 48, 51, 56, 64, 71, 73], которые отличаются способами управления ДРМ, количеством контроллеров и ДРМ в составе одной ДРС, организацией памяти конфигураций для ДРМ, подсистемой взаимодействия с центральным вычислителем (host computer), областью применения и т.д. В настоящей главе приведен обзор как базовых, так и более сложных архитектурных решений для динамически реконфигурируемых систем обработки информации.

2.1. Адаптивное вычислительное устройство

Современные темпы развития вычислительной техники характеризует закон Мура. Теоретически не доказанный, этот закон, первоначально предсказавший удвоение реализуемой степени интеграции компонентов внутри чипов интегральных схем каждые полтора года, оказался приложим и к росту производительности ЭВМ [28]. Несмотря на явные успехи, развитие вычислительной техники движется вперед поступательно, во многом следуя классическим постулатам. Один из них − принцип программного управления, базовые идеи которого были сформулированы, например, Джоном фон Нейманом [9] и Дж. Айлифом [1].

Прогресс в технологии ДР ПЛИС позволил перейти к проектированию адаптивных вычислительных устройств (АВУ), способных менять свою внутреннюю структуру в зависимости от поставленной задачи [33]. Цель создания АВУ − повышение эффективности обработки данных, что особенно актуально для мультимедийных потоков информации [33]. Критерием эффективности обработки данных может быть комплексная оценка производительности, программируемости и потребляемой мощности (Performance, Programmability and Power – 3P’s) [68].

Структурная адаптация как средство повышения производительности – новое явление для вычислительной техники, поэтому основ-

ная задача исследований (как теоретических, так и экспериментальных) состоит в выявлении преимуществ и недостатков предлагаемых устройств. Важное и интересное свойство АВУ − универсальность, т.е. возможность решения любой вычислительной задачи. Подобным свойством до появления ДР ПЛИС обладали только процессоры. Однако различия между универсальностью процессоров и АВУ велики. Заключаются они в базовых принципах построения устройств [33].

В основе всех процессорных устройств лежит принцип программного управления. В подавляющем большинстве случае это – фон-неймановский вариант, который состоит в следующем [9, 23]:

1) информация кодируется в двоичной форме и разделяется на единицы (элементы) информации, называемые словами;

2) разнотипные слова информации различаются по способу использования, но не способами кодирования;

3) слова информации размещаются в ячейках памяти машины и идентифицируются номерами ячеек, называемыми адресами слов;

4) алгоритм представляется в форме последовательности управляющих слов, которые определяют наименование операции и слова данных, участвующие в операции. Они называются командами. Алгоритм, представленный в терминах машинных команд, называется программой;

5) выполнение вычислений, предписанных алгоритмом, сводится к последовательному выполнению команд в порядке, однозначно определяемом программой.

Главный недостаток фон-неймановского принципа − ограниченность и фиксированность набора операций вычислительного устройства. Действительно, число известных на сегодня методов расчета различных функций достаточно велико, а число вычислительных задач вообще бесконечно. В таком случае справедливо утверждение, что оптимальная реализация всех алгоритмов не возможна посредством ограниченного и фиксированного набора вычислительных операций [33].

Очевидное следствие этого – проблемы, связанные с синтезом системы команд и построением высокопроизводительных устройств.

Сейчас для их эффективного решения применяются специализированные процессоры, система команд которых является проблемно-ориентированной [38]. Подобный подход позволяет реализовывать многие типовые фрагменты алгоритмов аппаратно, что положительно сказывается на производительности устройств. В специализированных процессорах, как и в любых других, операции инициируются посредством команд, таким же образом определяются адреса операндов [23]. Набор операций не может быть скорректирован без изменения или дополнения аппаратного обеспечения [33].

В этом отношении АВУ – значительно более гибкая система. В ней можно выделить два основных элемента (рис. 2.1) [33]:

- микропрограммный автомат (контроллер АВУ),
- вычислитель (ДР ПЛИС).

Рис. 2.1. Обобщенная структура АВУ и распределение потока данных в нем

Микропрограммный автомат базируется на принципах программного управления и использует следующие типы операций [33]:

– посылочные (применяются для пересылки информации между основной памятью системы и АВУ);

– загрузочные (предназначены для загрузки конфигураций; представляют собой особый вид посылочных операций, их можно выделить в отдельный тип из-за специфичности выполняемых задач);

– ввода-вывода (служат для передачи информации между основной памятью и внешними устройствами системы) [23];

– системные (предназначаются для управления режимами работы системы) [23].

Можно заметить, что в списке отсутствуют арифметические и логические операции, а также операции перехода. Они задаются посредством изменения структуры вычислителя, позволяя для каждого алгоритма использовать оптимальный способ обработки информации. Из этого следует, что после загрузки конфигурации и задания параметров данные в АВУ обрабатываются без использования программного управления. При этом главная задача контроллера – организовать поток данных между вычислителем и основной памятью системы. Фактически она сводится к определению значений (адресов) соответствующих операндов и последующей их пересылке в ДР ПЛИС [33].

Такое разделение функций между вычислителем и микропрограммным автоматом связано с тем, что загрузочные, посылочные, ввода-вывода и системные операции инвариантны по отношению к решаемым задачам – их реализация зависит только от особенностей вычислительной системы, в которой используется АВУ (организации памяти, интерфейса ввода-вывода и т.п.). Следовательно, общую численность таких операций можно фиксировать, а алгоритмы их выполнения – оптимизировать под особенности системы. При построении контроллера АВУ целесообразно использовать программный принцип управления, реализуя перечисленные типы операций как команды [33].

Арифметические и логические операции, а также операции перехода, напротив, должны отражать алгоритм поставленной задачи, поэтому ограниченность набора вычислительных операций неизбежно ведет к тому, что только определенные классы задач будут оптимизи-

рованы под структуру АЛУ. Для устранения указанных недостатков применяются ДР ПЛИС, способные аппаратно реализовать алгоритмы любых вычислительных задач. В то же время следует отметить, что операции перехода в ряде случаев могут зависеть не только от алгоритма, но и от особенностей вычислительной системы (например, от способа хранения программы, логической емкости ДР ПЛИС), поэтому для повышения эффективности АВУ можно предусмотреть и программный способ реализации данного типа операций. Примеры наиболее характерных ситуаций, которые могут возникнуть при реализации операций перехода, приведены на рис. 2.2, где У1 – операции перехода, В1, В2, В3 – арифметические и логические операции [33].

Другим фактором, позволяющим повысить производительность устройства, является то, что в вычислителе реализуются довольно крупные фрагменты алгоритмов, представляющие собой композиции элементарных операций. Известно, что аппаратные средства характеризуются отсутствием промежуточных пересылок информации в процессе выполнения каждой операции [9]. Таким образом, в АВУ уменьшается число избыточных обращений к памяти, а значит, сокращается общее время вычислений. Дальнейшее увеличение быстродействия устройства в этом направлении связано с введением кэш-памяти (рис. 2.3) [33].

Все рассмотренное выше указывает на то, что АВУ основывается на принципе, отличном от предложенного Джоном фон Нейманом. Еще одним доказательством этого служит различие в формах представления алгоритма в адаптивных и процессорных системах. В АВУ алгоритм описывается совокупностью команд, обрабатываемых контроллером, и конфигурационных данных, задающих (определяющих) внутреннюю структуру вычислителя. Их нельзя считать эквивалентными друг другу, по крайней мере, по двум причинам [33]:

– команда несет информацию не о структуре процессора, а о его функциональных возможностях;

– конфигурационная информация описывает структуру вычислительного устройства, но не обрабатываемые данные.

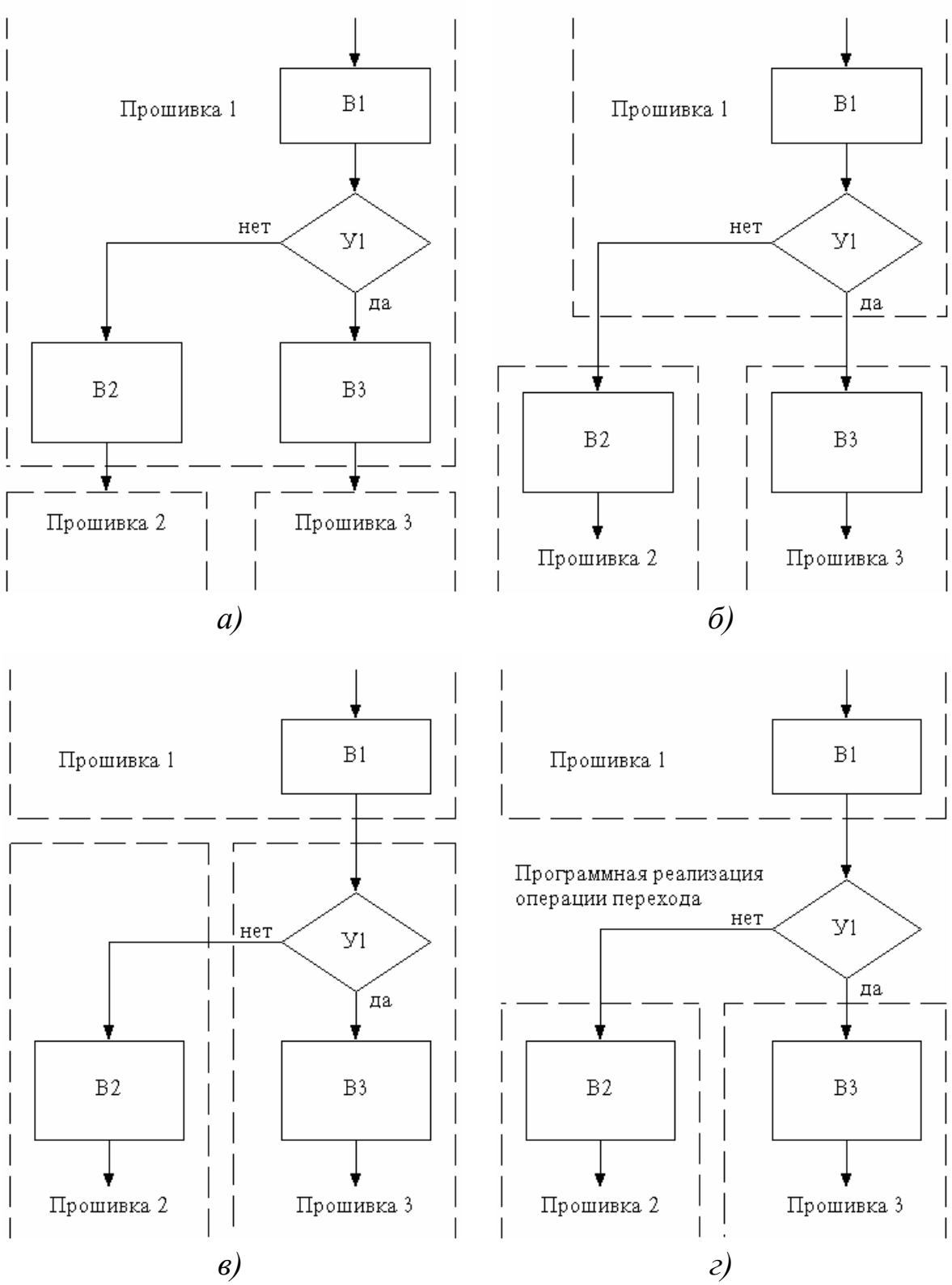


Рис. 2.2. Примеры реализации операций перехода:
а − аппаратная реализация, не требующая операций со стороны контроллера АВУ; б − аппаратная реализация, требующая загрузки конфигурационной информации по признакам перехода; в − неэффективная аппаратная реализация; г − программная реализация

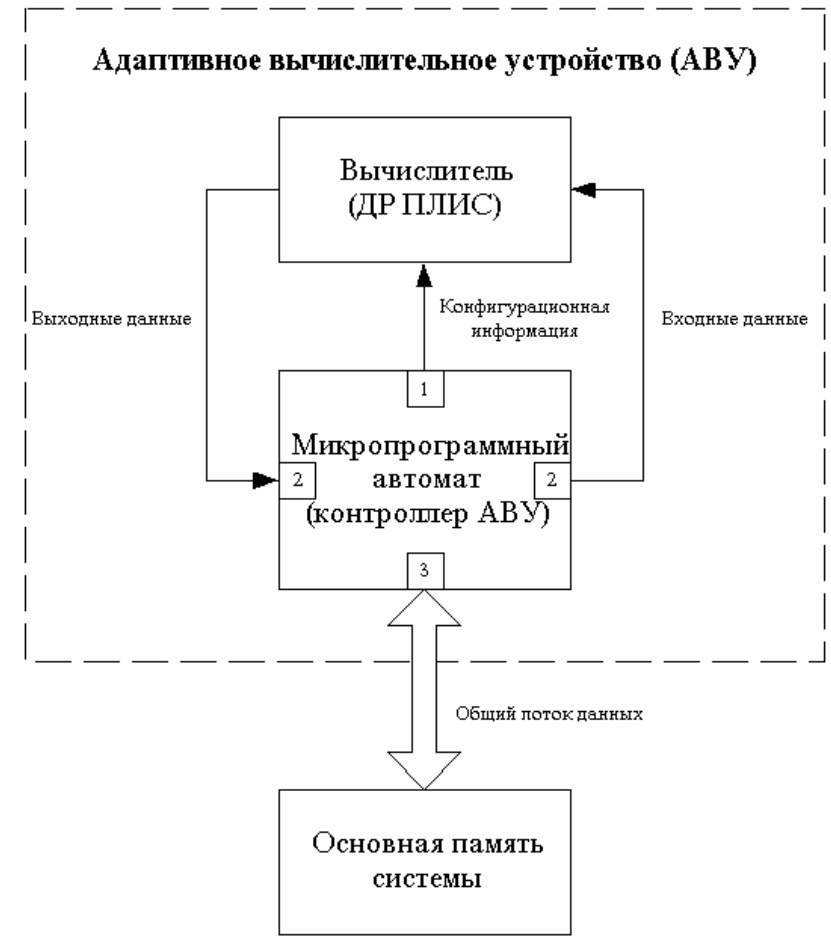


Рис. 2.3. Использование кэш-памяти в АВУ:
1 – кэш-память для конфигурационной информации;
2 – кэш-память для входных и выходных данных;
3 – кэш-память для команд контроллера АВУ

Операции в контроллере и вычислителе выполняются одновременно и независимо [33].

По аналогии с гарвардской архитектурой [37] можно предложить разделение памяти на три типа, что позволит обращаться одновременно к различным устройствам памяти (рис. 2.4) [33]:

- память программ (ПП),
- память конфигураций (ПК),
- память данных (ПД).

Стратегически важная область применения, в которой адаптивные системы потенциально способны вполне успешно конкурировать с процессорами, − проектирование высокопроизводительных систем. Действительно, алгоритм любой задачи может быть представлен как

композиция параллельно и последовательно выполняемых операций, причем виды параллелизма могут отличаться для отдельных его участков. Для достижения максимальной пользовательской производительности необходимо устройство обработки информации, которое могло бы производить вычисления без дополнительных преобразований одних видов параллелизма в другие. Создание вычислительных устройств с такими свойствами реально только для адаптивных систем [33].

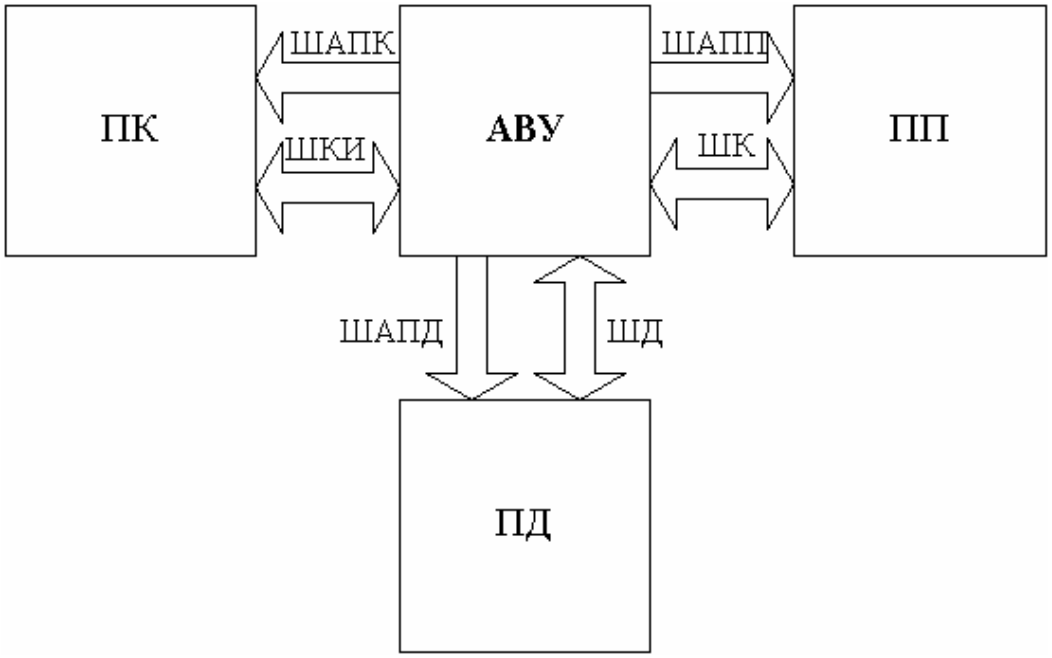


Рис. 2.4. Гарвардская архитектура АВУ:
ШАПД – шина адреса ПД, ШАПК – шина адреса ПК, ШАПП – шина адреса ПП, ШД – шина данных, ШК – шина команд,
ШКИ – шина конфигурационной информации

2.2. Комплементарное вычислительное устройство

Как следует из определения ДРС, их главный недостаток, ограничивающий область применения, состоит в том, что временные затраты на реконфигурирование могут превышать тот выигрыш во времени, который был или будет получен благодаря адаптивности системы команд ДРС. В частности, как следует из математической модели процесса реконфигурирования, предложенной в [30, 33] для уст-

ройств на базе ПЛИС, ДРС не целесообразно использовать для выполнения малого числа (в предельном случае единичных) операций. Для устранения указанного недостатка можно использовать так называемые «вычислители с гибридной архитектурой» (hybrid-architecture computers), к которым относятся все ДРС, состоящие из универсального процессора и динамически реконфигурируемой ПЛИС. В настоящей работе рассматривается концепция построения одного из вариантов гибридных ДРС – комплементарного вычислительного устройства [43], которое состоит из:

– адаптивного вычислительного устройства, подробно рассмотренного в [33];

– процессора, базирующегося на принципах программного управления;

– устройства контроля за доступом к памяти (рис. 2.5).

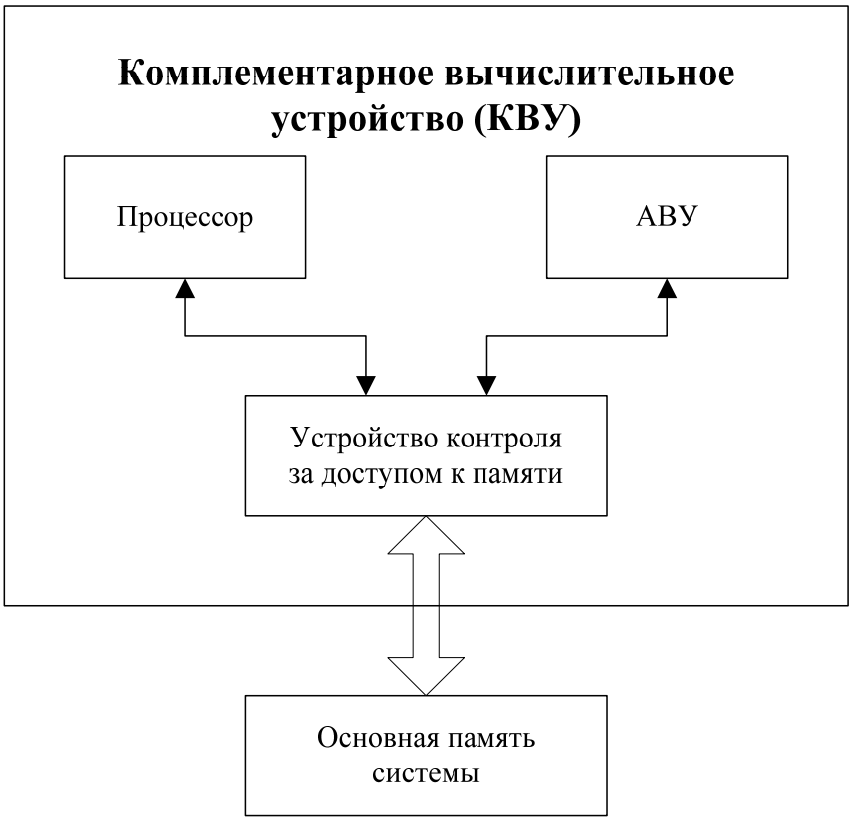


Рис. 2.5. Обобщенная структурная схема КВУ

Принцип работы КВУ целесообразно рассмотреть на примере расчета среднеквадратического отклонения:

$$\sigma = \frac{\sqrt{\sum_{i=1}^{N}(x_i - \overline{x})}}{N-1}.$$

Вычисление разницы между *i*–м и средним значениями с накоплением полученного результата повторяется *N* раз. При $N \geq N_{кр}$ (где $N_{кр}$ – число операций без реконфигурирования, при котором выполняются условия эффективности ДРС, предложенные в [30]), время расчета с использованием ДРС будет меньше, чем при программной реализации. Однако расчет квадратного корня и деление полученного результата на *N*–1 производятся однократно. Следовательно, реализация такого малого числа операций на базе АВУ будет неэффективной [43].

Вообще, на практике можно найти множество задач, требующих применения вычислительной техники, где наряду с обработкой массивов данных и циклически повторяемыми расчетами необходимо производить выполнение единичных операций. Таким образом, в КВУ для выполнения многократно повторяющихся математических операций используется АВУ, а для выполнения единичных и большинства нематематических операций – процессор. Устройство контроля за доступом к памяти предназначено для предотвращения одновременного обращения АВУ и процессора к памяти, а также первичной обработки (сортировки) команд. В одном из полей команд КВУ указывается код устройства, которое будет выполнять операцию – АВУ или процессор, а устройство контроля за доступом к памяти обеспечивает пересылку команды к соответствующему вычислительному узлу. Для КВУ, как и для АВУ, возможна гарвардская архитектура (рис. 2.6). В этом случае используются четыре вида памяти [43]:

– память данных, общая как для АВУ, так и для процессора;

– память программ процессора (ППП), используемая для хранения программ, выполняемых процессором;

– память программ АВУ (ПП АВУ), используемая для хранения программ, выполняемых контроллером АВУ;

– память конфигураций АВУ (ПК АВУ), используемая для хранения конфигураций АВУ.

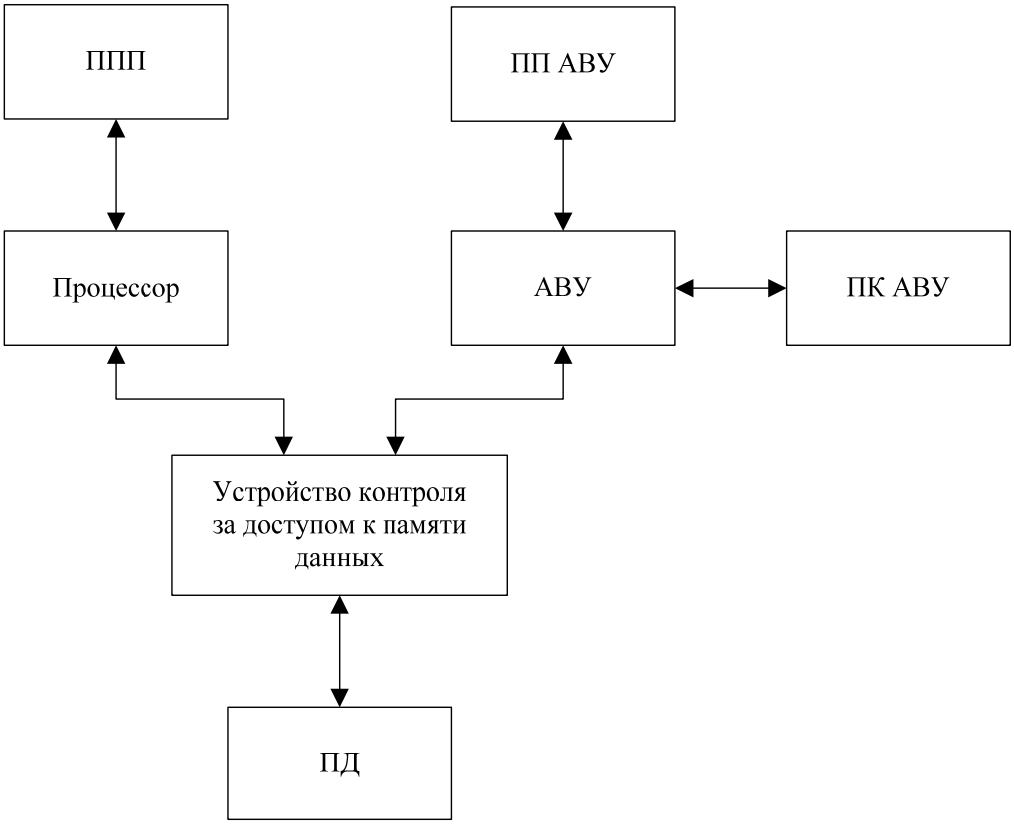


Рис. 2.6. Гарвардская архитектура КВУ

Это позволяет естественным образом снизить нагрузку на каналы «память – устройство обработки данных» и увеличить производительность вычислительной системы. При использовании гарвардской архитектуры устройство контроля за доступом к памяти необходимо только для предотвращения одновременного обращения АВУ и процессора к общей памяти данных. Дальнейшее повышение производительности КВУ возможно за счет введения кэш-памяти, увеличения числа модулей обработки данных (процессоров и АВУ), конвейеризации вычислительных модулей, использования различных архитектур процессоров (матричной, векторной, скалярной, суперскалярной и т.д.) [43].

Разработка программного обеспечения для системы, устройством обработки данных в котором является КВУ, значительно отличается от разработки программного обеспечения для классических процессорных систем. Это связано с тем, что данные могут обрабатываться процессором, контроллером АВУ и вычислителем АВУ. При этом директивы

процессору и контроллеру АВУ представляют собой команды, а директивы вычислителю АВУ – конфигурационные файлы. Подобная специфика требует разработки нового методического, лингвистического и математического обеспечения, а также новых сред программирования, что можно считать основным недостатком КВУ [43].

Предложенная гибридная ДРС может применяться в ЭВС различной производительности (от вычислителей малого быстродействия до высокопроизводительных вычислительных систем). Действительно, алгоритм любой задачи может быть представлен как композиция различных видов параллелизма и последовательно выполняемых операций. Для достижения максимальной пользовательской производительности необходимо устройство обработки информации, которое могло бы производить вычисления без дополнительных преобразований одних видов параллелизма в другие [33]. Создание вычислительных устройств с такими свойствами вполне реально для КВУ [43].

2.3. Обзор базовых архитектурных решений для динамически реконфигурируемых систем обработки информации

АВУ и КВУ − теоретическое обобщение тех практически значимых решений, которые применялись в вычислительных системах, начиная с 90-х годов XX века. Главная их цель – научно и максимально формально объяснить те потрясающие экспериментальные результаты, которые были достигнуты с использованием ДРС самых различных архитектур. В настоящем разделе представлен обзор архитектур, применяемых в промышленно выпускаемых изделиях и опытных образцах. Он позволит перейти от теоретического анализа к описанию реальных вычислительных устройств.

Наиболее очевидное применение ДР ПЛИС − так называемый динамически реконфигурируемый вычислитель (ДРВ) [31]. Он состоит из универсального малоразрядного микропроцессора, выполняющего управляющие операции, и реконфигурируемого сопроцессора (РСП), выполненного на ПЛИС и реализующего обработку данных

(рис. 2.7). Аппаратные ресурсы сопроцессора позволяют быстро выполнять одну или несколько операций, специфичных для данной системы. Когда в процессе выполнения программы возникает необходимость вычисления следующей функции, производится реконфигурация ПЛИС в соответствии с требуемой операцией. Для реализации динамического реконфигурирования в систему вводятся также память конфигураций и схема управления загрузкой конфигурации [31].

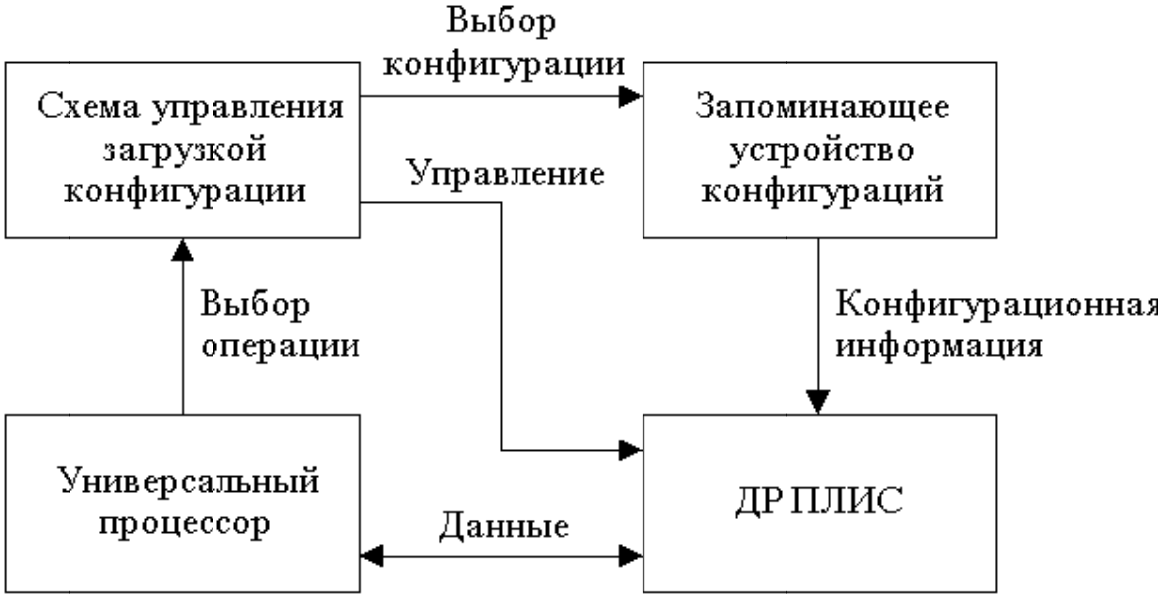


Рис. 2.7. Структурная схема ДРВ

Поскольку в каждый момент времени от вычислителя требуется возможность производить ограниченное число операций (в предельном случае − одну), для реализации сопроцессора можно применить дешевую ПЛИС с относительно малой логической емкостью. Это не накладывает ограничений на число операций в наборе, доступном для выполнения сопроцессором (ограничения накладывает лишь объем памяти, требуемый для хранения конфигурационных данных). Использование ПЛИС с небольшой логической емкостью позволяет также уменьшить требуемый объем ЗУ конфигураций, а также время, требуемое для пересылки этой информации в ПЛИС. Схема управления загрузкой конфигурации служит для обеспечения параллельной работы процессора и загрузки конфигурации в РСП. Если параллельная работа не требуется, то управление может быть реализовано про-

граммно в процессоре, а память конфигурации совмещена с основной системной памятью. Загружаемая в ПЛИС логическая схема состоит из двух частей [31]:

1) интерфейса, обеспечивающего взаимодействие с необходимым типом процессора (он является уникальным для каждого семейства микропроцессоров и может быть загружен в ПЛИС один раз при инициализации системы);

2) вычислителя, выполняющего собственно обработку данных (он может выполняться независимым от типа процессора).

Цикл выполнения операции в простейшем виде выглядит следующим образом:

1) определение требуемой функции;
2) загрузка конфигурационной памяти для требуемой функции;
3) загрузка исходных данных;
4) вычисление;
5) считывание результата.

Первый этап выполняется микропроцессором (ведущим процессором) в соответствии с алгоритмом функционирования системы. По его завершению становится известной требуемая конфигурация ПЛИС. Если она отличается от текущей, то необходимо реконфигурирование (этап 2). После этого возможна собственно обработка данных (этапы 3 − 5) [31].

ДРВ может быть реализован как на базе ДР ПЛИС как с полной, так и с частичной реконфигурацией. При использовании ПЛИС первого типа (например, Altera ACEX) в каждый момент времени возможно выполнение только одной операции и невозможно функционирование ПЛИС во время загрузки конфигурации. Поэтому наиболее критично время конфигурирования. Целесообразно организовать обработку данных так, чтобы минимизировать число реконфигураций (например, подготавливать массивы данных). Также необходимо загружать в ПЛИС каждый раз неизменную интерфейсную часть, что увеличивает объем конфигурационных данных. Преимущества таких ПЛИС − их более простая структура и, как следствие, низкая стоимость. Если ПЛИС поддерживает частичную реконфигурацию

(например, Atmel AT40K), то возможно выполнение различных операций параллельно в различных участках ПЛИС, например:

Участок 1: вычисление функции A,

Участок 2: вычисление функции B,

Участок 3: конфигурирование для вычисления функции C,

Участок 4: загрузка данных.

При этом также допустимо использование части логической емкости ПЛИС не для организации вычислений, а в качестве самостоятельной логической схемы (интерфейса и т.п.) [31].

Очевидно, что ДРВ − альтернатива современным микроконтроллерам и цифровым сигнальным процессорам. Принципиальное отличие от дешевых малоразрядных микроконтроллеров состоит в том, что ДРВ обладает дополнительными возможностями из−за наличия математического РСП. При сравнении с цифровыми сигнальными процессорами к основным достоинствам ДРВ можно отнести невысокую себестоимость и практически неограниченный набор вычислительных операций. Основные области применения ДРВ − системы управления станками с ЧПУ, модули обработки мультимедийной информации, т.е. такие устройства, в которых необходимо производить значительный объем расчетов без обращения к центральной ЭВМ.

Функциональные возможности ДРВ могут быть расширены за счет применения нескольких РСП и более производительного микропроцессора. Взаимодействие между сопроцессорами зависит от архитектуры ДРВ, связанной с управлением РСП и числом ЗУ конфигураций. В случае реализации данного подхода целесообразно разрабатывать конфигурации, ориентировать на различные модели параллелизма (естественный параллелизм, параллелизм множества объектов, параллелизм независимых ветвей, параллелизм смежных операций, смешенный параллелизм и т.д.). При этом в ДРВ возможна специализация сопроцессоров по определенным видам параллелизма, что приводит к необходимости использования нескольких ПЛИС в одном устройстве.

Пример практической реализации ДРВ был рассмотрен в [20]. На рис. 2.8 показан лабораторный стенд, описанный в [20] и внедренный

в учебный процесс на кафедре конструирования и технологии радиоэлектронных средств Владимирского государственного университета в составе аппаратно-программного комплекса ACEXLab 2.3 [52].

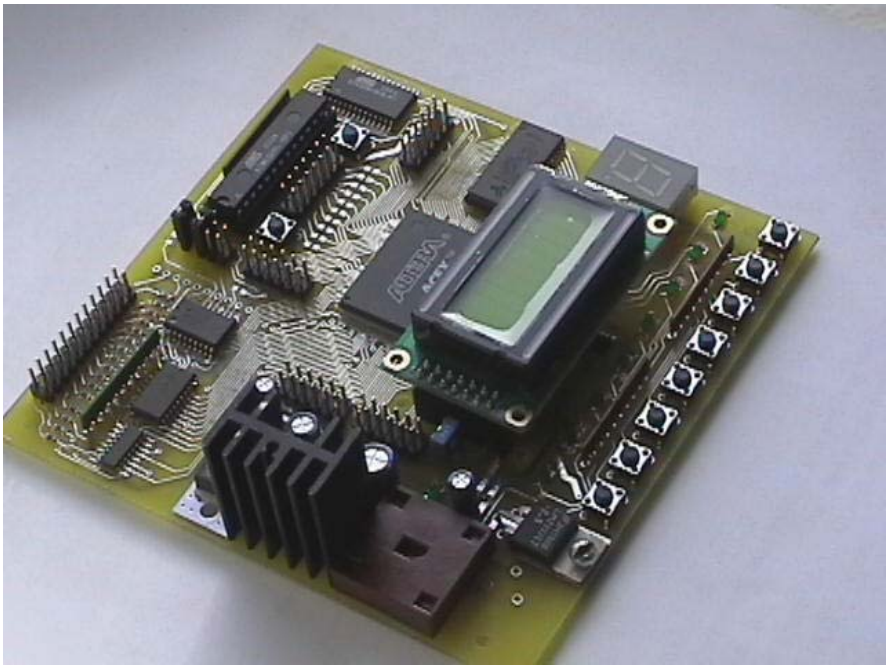


Рис. 2.8. Лабораторный стенд, состоящий из вычислительной системы типа ДРВ и различных устройств ввода-вывода

Технические характеристики стенда перечислены в табл. 2.1.

Таблица 2.1

Технические характеристики лабораторного стенда, построенного по архитектуре ДРВ

Элемент	Описание
Микроконтроллер	ATMEGA8L-8PI
ДР ПЛИС	EP1K100 (Altera ACEX1K)
Статическая ОЗУ с произвольным доступом (SRAM)	AS7C34096 (8×512 Кбит)
Перепрограммируемая Flash-память	AT45DB081B (1 Мбайт)
Периферийные устройства ввода/вывода	Порт LPT (EPP)

Другой простейший вариант ДРС − вычислитель с динамически изменяемой системой команд (dynamic instruction set computer –

DISC) [72]. DISC состоит из двух ДР ПЛИС, памяти конфигураций и персонального компьютера (рис. 2.9).

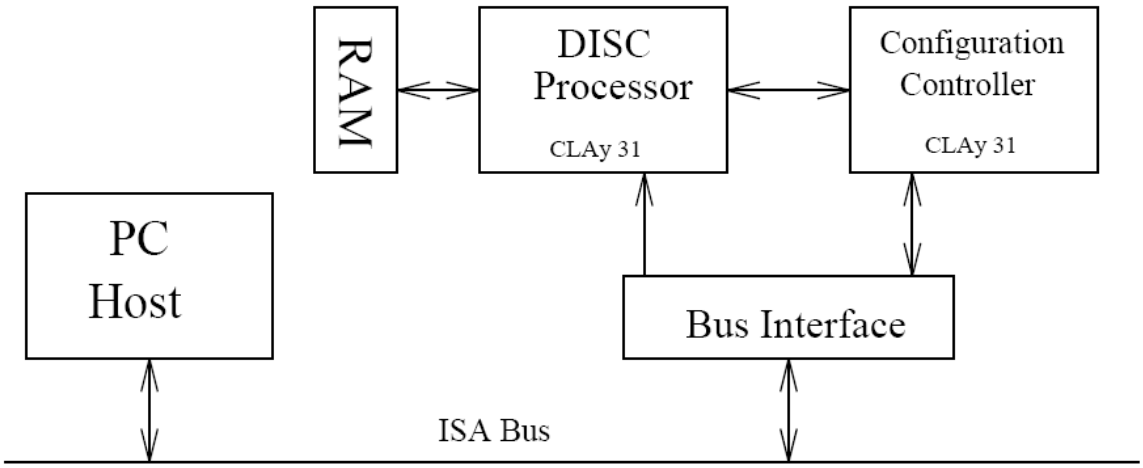


Рис. 2.9 Структурная схема электронно-вычислительного комплекса, использующего в своей работе вычислитель с динамически изменяемым набором команд

Первая ДР ПЛИС используется в качестве динамически реконфигурируемого процессора, вторая – это контроллер, который необходим для загрузки конфигурационных данных из памяти конфигураций в процессор. Обе ПЛИС относятся к семейству CLAy 31 фирмы National Semiconductors. Если в памяти конфигураций не оказывается нужной конфигурации, то она выбирается из тех, которые хранятся в ЗУ персонального компьютера [72].

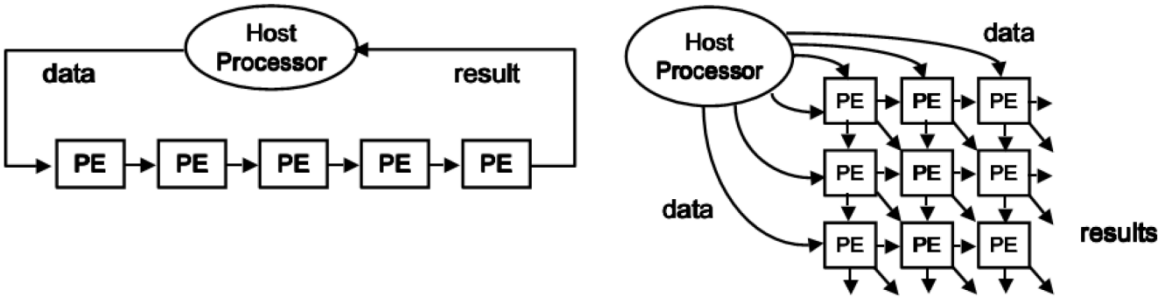
Потенциальная область применения DISC − устройства, осуществляющие интенсивный обмен данными с персональным компьютером. В первую очередь к ним относятся вычислительные устройства, реализованные в виде плат расширения (графические акселераторы, внутренние модемы и т.п.), подключаемые через локальные шины (в частности, через шину ISA – Industrial Standard Architecture [72]).

Наиболее существенными отличиями DISC от ДРВ:

– в структуре DISC присутствует не микропроцессор, а персональный компьютер;

– для хранения конфигураций используется не только специальное ЗУ конфигураций, но ЗУ персонального компьютера.

Идеи, аналогичные тем, которые были заложены в DISC, используются в ДРС, построенных на базе систолических архитектур SPLASH, SPLASH-II (рис. 2.10) и WILDFIRE [56, 64]. Основное отличие состоит в количественном увеличении числа ДРМ. Систолические архитектуры (см. рис. 2.10), которые представляют собой объединение большого числа простых вычислительных элементов, удачно подходят для реализации алгоритмов на базе ПЛИС, что обусловлено особенностями разработки конфигурационных файлов для этой элементной базы. Проектирование сложных вычислительных устройств на базе ПЛИС требует значительных временных затрат. Однако внутренняя архитектура ПЛИС такова, что позволяет достаточно просто реализовывать параллельные вычисления. Именно по этой причине систолические архитектуры широко используются в ДРС.



а) *б)*

Рис. 2.10. Примеры систолических архитектур [67]:
а – одномерная архитектура; б – двумерная архитектура;
PE – вычислительный элемент, Host Processor – центральный процессор,
data – данные, results – результаты

Следует подчеркнуть, что ДРС с систолической архитектурой наглядно демонстрируют многие достоинства таких вычислительных систем. В частности, контроллеры подобных ДРС не требуют реализации сложных алгоритмов управления, так как согласованность работы ДРМ обеспечивается самой систолической архитектурой. Кроме того, масштабирование таких ДРС не представляет большой сложности – достаточно просто нарастить число ДРМ. Это позволяет создавать вычислительные комплексы высокой производительности, обладающие, кроме всего прочего, способностью адаптироваться к решаемым задачам. В качестве примера подобного решения можно

привести архитектуру SPLASH-II, рассмотренную на рис. 2.11. Данный вычислитель состоит из вычислительных элементов (processing element), обозначенных на рисунке литерами «X» и представляющих собой ПЛИС XC4010 фирмы Xilinx с подключенной к ней оперативной памятью. Вычислительные элементы объединяются в вычислительные модули по 17 штук и дополнительно соединяются между собой через коммуникационный интерфейс (crossbar), который используется при решении специальных задач. Вычислительные модули через интерфейсный модуль подключаются к центральному компьютеру (Sun SPARC) [56].

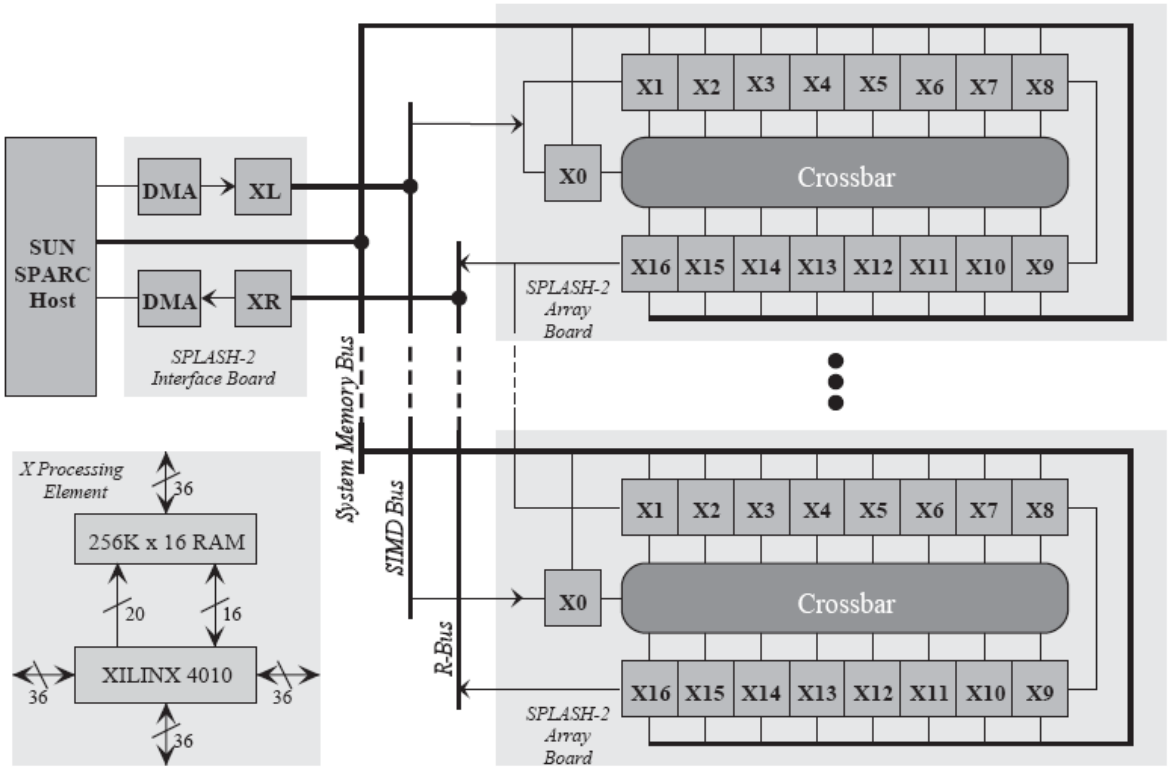


Рис. 2.11. Структурная схема ДРС, построенной по архитектуре SPLASH-II

Другая систолическая ДРС, базирующаяся на архитектуре SPLASH-II и на другой аналогичной архитектуре WILDFIRE [64], была реализована в виде PCI-модулей (PCI – Peripheral Component Interconnect) семейства WILDFORCE [64]. Основное сходство с DISC состоит в использовании локальной шины для обмена данными с компьютером, а главное отличие в том, что предлагаемое решение является масштабируемым.

На структурной схеме модулей семейства WILDFORCE (рис. 2.12) видно, что ДРС может включать до 5 вычислительных элементов (PE − processing element), которые состоят из ДР ПЛИС (семейств Xilinx XC4000-XL) и подключенной к ней локальной оперативной памяти.

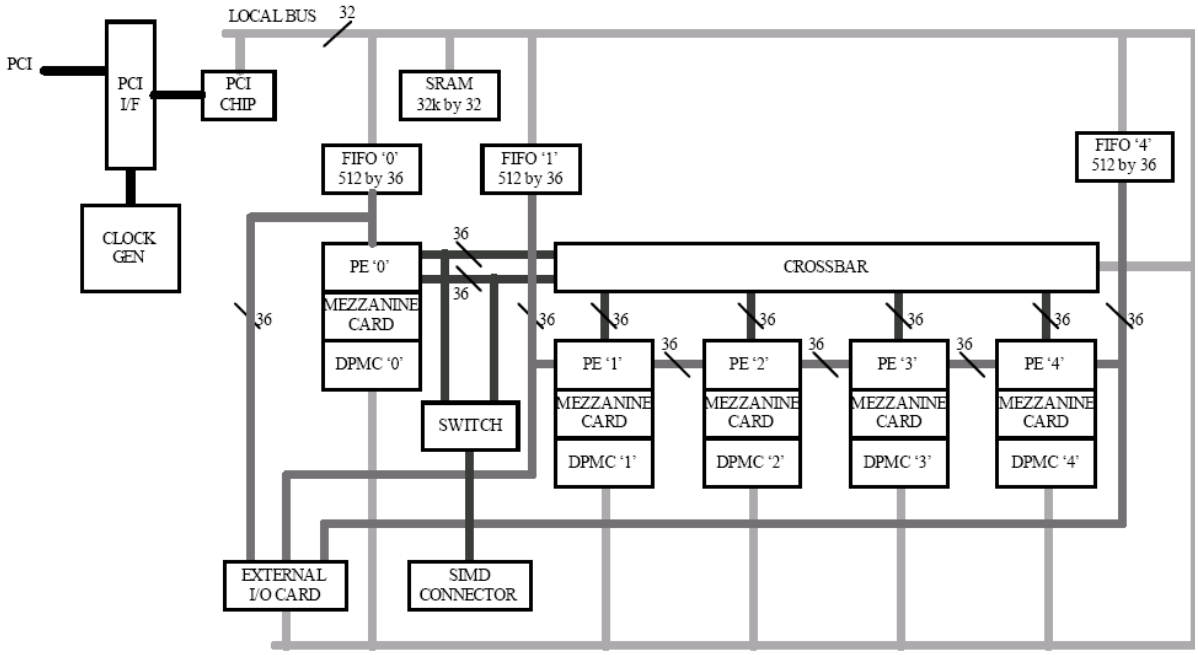


Рис. 2.12. Структурная схема PCI-модуля семейства WILDFORCE, построенного на базе ДРС

Обмен данными осуществляется как через систолические 36-разрядные двунаправленные шины, соединяющие соседние элементы, так и с использованием топологии типа «звезда», то есть через «свитч» (crossbar switch, CROSSBAR) [64]. Кроме того, на модуль интегрированы оперативная память (SRAM), FIFO-буферы, PCI-контроллер (PCI CHIP) и тактовый генератор для него (CLOCK GEN), а также предусмотрена возможность подключения дополнительных плат ввода-вывода (EXTERNAL I/O CARD) и других устройств через 36-разрядный разъем SIMD (SIMD CONNECTOR) [64]. Подобные устройства использовались в качестве ускорителей при решении задач цифровой обработки сигналов (в частности, для цифровой фильтрации) и были протестированы на различных аппаратных (персональных компьютерах IBM PC, рабочих станциях DEC Alpha, серверах Silicon Graphics Origin-200 и Origin-2000) и программных (операционных сис-

темах Microsoft Windows, UNIX, IRIX) платформах. Положительным свойством таких плат является то, что содержимое оперативной памяти каждого вычислительного модуля непосредственно через шину PCI и DPMC (Dual-Port Memory Controller) доступно для компьютера.

Конструктивно PCI-модули семейства WILDFORCE выполнены в виде материнской платы, параллельно которой подключаются дочерние платы (MEZZNINE CARD), реализующие функции вычислительного элемента (рис. 2.13). Сочетание подобных архитектурных, схемотехнических и конструктивных решений обеспечивает простоту масштабирования внутри динамически реконфигурируемой системы обработки информации. Подробно архитектурные особенности и технические характеристики PCI-модулей семейства WILDFORCE рассмотрены в [64].

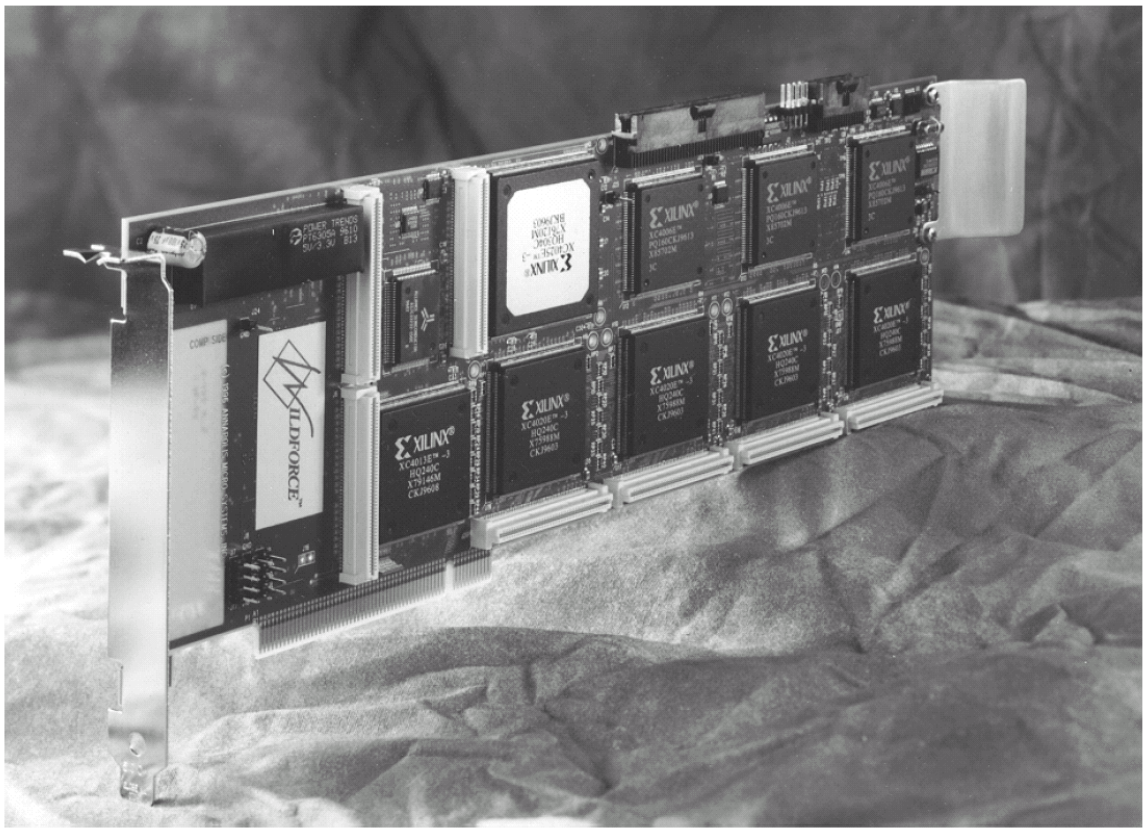


Рис. 2.13. PCI-модуль семейства WILDFORCE

Другой вариант вычислителя (рис. 2.14), имеющий сходство с ДРВ и DISC, называется Programmable Active Memories (PAM), что можно перевести как «программируемые активные запоминающие

устройства». Он также построен на базе ДР ПЛИС и является виртуальной машиной, управляемой обычным микропроцессором, который динамически конфигурирует массив ПЛИС, каждая из которых рассматривается как заказная микросхема, способная менять свое функциональное назначение с течением времени [71]. По периметру к массиву ПЛИС подключены модули оперативной памяти [71]. В отличие от DISC и PCI-модулей семейства WILDFORCE, которые подключались к компьютеру соответственно через локальные шины ISA и PCI, PAM для обмена данными использует системную шину [71], что позволяет существенно увеличить скорость приема-передачи данных.

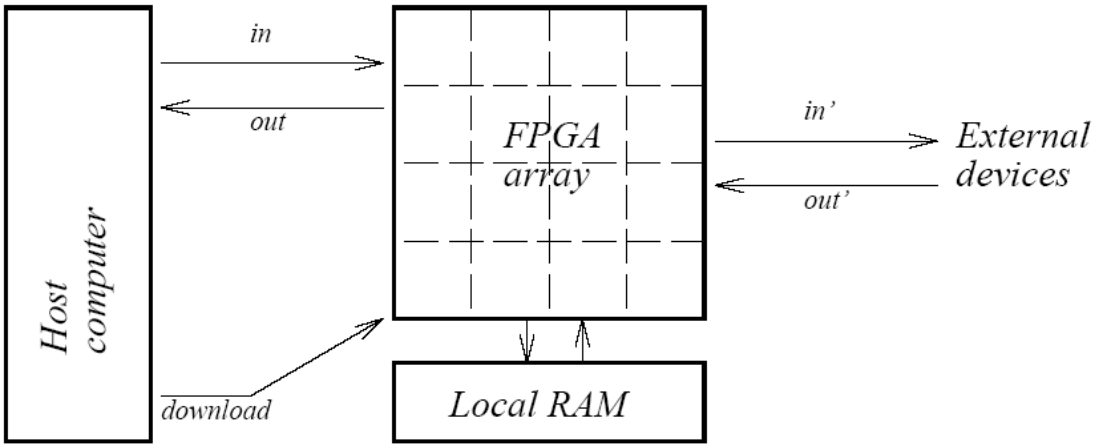


Рис. 2.14. Обобщенная структурная схема PAM

Подобная архитектура нашла применение при реализации нейронных сетей, алгоритмов двумерной свертки, алгоритмов компрессии, анализа и синтеза звуковых сигналов, изображений и видеоданных, криптографических систем, численных методов решения уравнений, а также при решении задач из области физики высоких энергий, термодинамики, астрономии, трехмерного моделирования и стереовидения [71].

На рис. 2.15 показана структурная схема вычислительного комплекса DEC PeRLe-1, построенного на основе архитектуры PAM. Данный комплекс предназначен для решения задач, обладающих значительной вычислительной сложностью. Конструктивно комплекс (рис. 2.16) состоит из трех элементов:

− основного вычислительного узла, построенного на основе рассматриваемой архитектуры ДРС;

− подсистемы ввода/вывода, использующей так называемые TURBOchannels;

− внешнего вычислительного модуля, выполняющего вспомогательные функции.

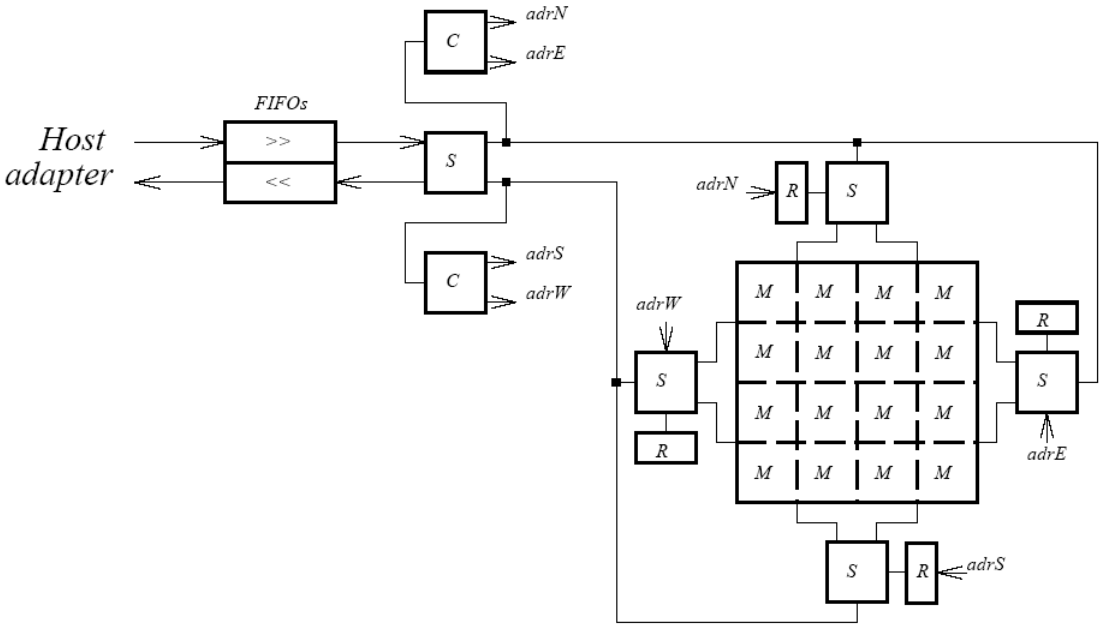


Рис. 2.15. Структурная схема вычислительного комплекса DEC PeRLe-1

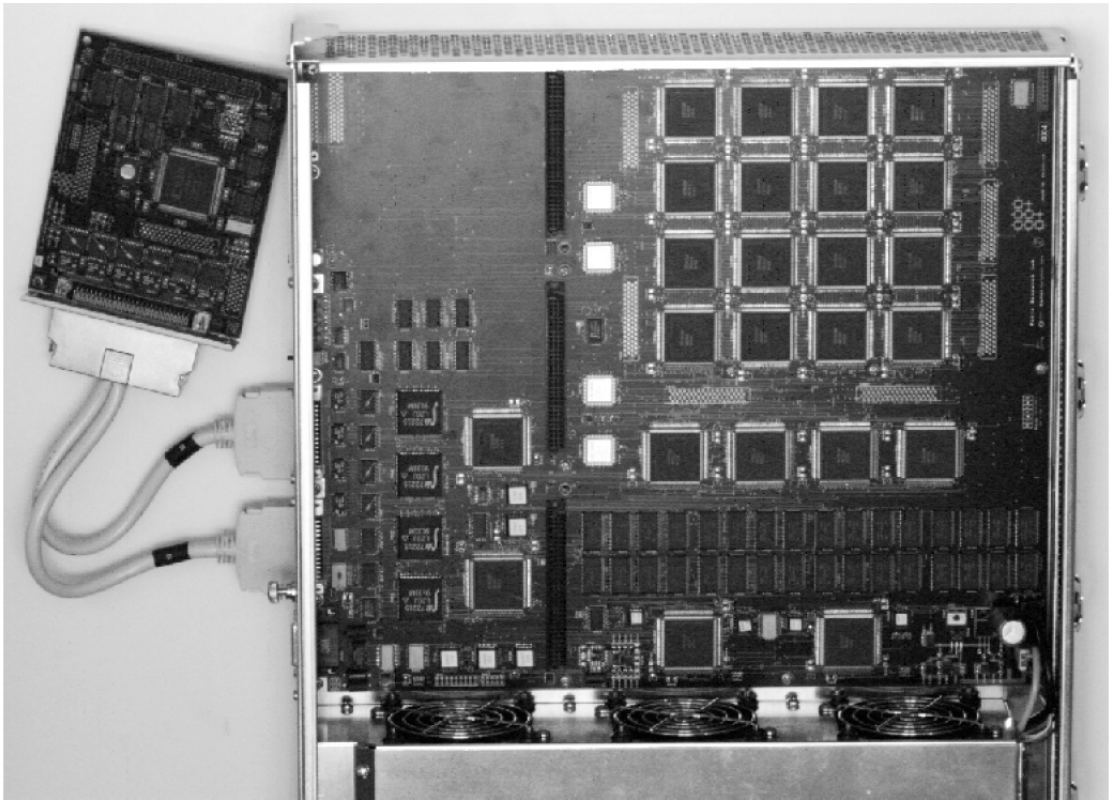


Рис. 2.16. Электронно-вычислительный комплекс DEC PeRLe-1

Очевидно, что существуют или есть возможность создать и другие подобные архитектуры ДРС, которые будут отличаться производительностью, потребляемой мощностью, применяемой элементной базой, сложностью масштабирования, доступностью и уровнем инструментов (низко- и высокоуровневые), предназначенных для реализации конечных приложений. Однако все они будут основаны на одном и том же принципе: динамически реконфигурируемый элемент используется в качестве вспомогательного вычислительного узла, не принимающего никакого участия в задаче диспетчеризации, то есть в распределении вычислительных ресурсов системы между имеющимися задачами. Данная функция полностью возложена на основной вычислитель (host computer), которым в случае ДРВ является микропроцессор, а в случае DISC, SPLASH и PAM – персональный компьютер.

Если принять в качестве критерия, определяющего зависимость одного вычислительного узла системы от другого, тот факт, где и каким образом распределяются вычислительные ресурсы между задачами, то особенно показательной архитектурой можно считать ту, которая впервые была предложена в [13]. Основной целью ее создания было разработка архитектуры ДРС, ориентированной на решение задач цифровой обработки сигналов (в частности, в области телекоммуникаций и связи, а также обработки биоэлектрических сигналов), которые принадлежат к числу наиболее перспективных направлений для применения ДРС.

Указанный вычислитель (рис. 2.17) относится к классу гетерогенных и состоит из:

− ***RISC-процессора***, выполняющего главным образом функции управления (в том числе распределяет вычислительные ресурсы и управляет загрузкой конфигурационных файлов в другие вычислительные узлы) и взаимодействия с внешними устройствами по соответствующим интерфейсам и протоколам, а также распределяющего задачи между цифровым сигнальным процессором и ПЛИС;

− ***ДР ПЛИС***, ориентированной на выполнение математических преобразований, содержащих большое число параллельных операций;

− ***цифрового сигнального процессора*** (ЦСП, процессор цифровой обработки сигналов, ПЦОС, digital signal processor, DSP), который работает на тактовой частоте, значительно превышающей аналогичный показатель для ПЛИС, и способен выполнять малое число операций (в частности, MAC-операций, то есть умножение с накоплением) за значительно меньшее время, чем это может делать ПЛИС [13, 51].

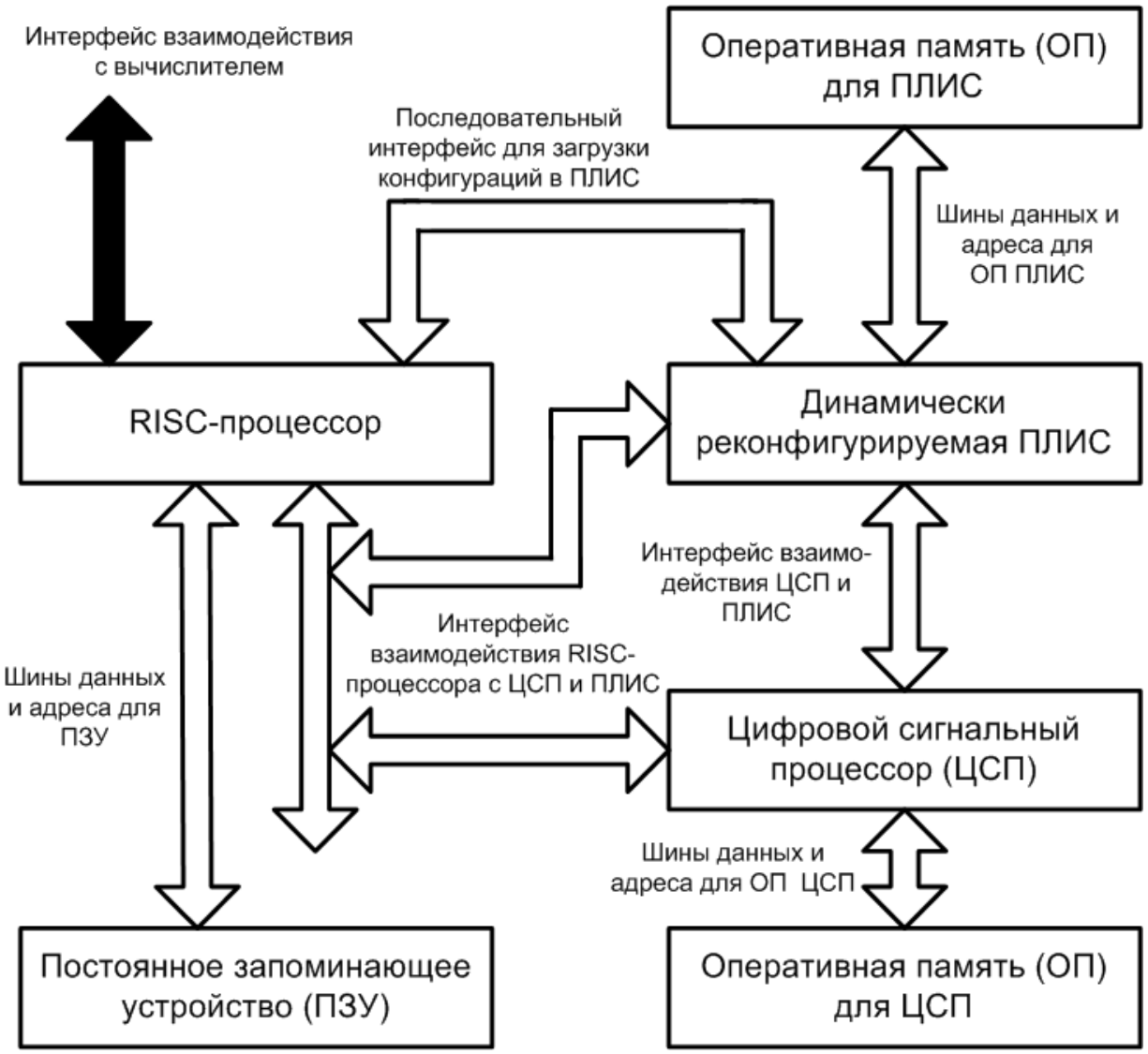


Рис. 2.17. Структурная схема гетерогенного вычислителя

В данном вычислителе процессоры связаны между собой по межпроцессорному интерфейсу. Таким образом, RISC-процессор выступает в качестве центрального («ведущего», master, host-processor), а ЦСП – в качестве «ведомого» (slave-processor). Например, если применяется ЦСП фирмы Texas Instruments Inc., то для указанных целей можно задействовать HPI-интерфейс (Host-Port Interface), разрядность которого составляет 16 или 32 бита. В этом случае для подключения ЦСП к RISC-процессору можно использовать шину памяти. Аналогичный подход допустим и для ДР ПЛИС, которая также подключается к шине памяти RISC-процессора. В свою очередь, для взаимодействия ЦСП компании Texas Instruments Inc. и ДР ПЛИС может применяться VLYNQ-интерфейс, который поддержан ведущими фирмами-производителями микросхем программируемой логики (например, Altera Corporation и Xilinx Inc.). Загрузка конфигурационных файлов в ЦСП и ДР ПЛИС осуществляется RISC-процессором, который выбирает данные или из постоянного запоминающего устройства, построенного по Flash- или FRAM-технологии, или из внешнего источника. При выборе RISC-платформы целесообразно ограничиться наиболее распространенными ядрами, такими, как ARM (прежде всего, ARM9, ARM11) и MIPS (например MIPS32) [51].

Очевидно, что при увеличении сложности ДРС возрастает и сложность их математических моделей. Это, в первую очередь, относится к гибридным вычислителям, которые могут состоять не только из большого числа ДР ПЛИС, но и включать в себя мультипроцессорные модули, построенные одновременно на базе универсальных, проблемно-ориентированных и специализированных процессоров. Кроме того, взаимосвязи между элементами такой вычислительной системы могут меняться с течением времени, поскольку ПЛИС способны выступать и в качестве коммутационных узлов, динамически определяющих направленность и тип линий передачи цифровых данных, а также ряд других характеристик. Все это лишний раз подтверждает тот факт, что основные проблемы математического моделирования современных ДРС заключаются не столько

в расчете времени их реконфигурации, сколько в учете параллелизма выполняемых операций. Это обусловлено важным достоинством ДРС, которое отличает их от традиционных вычислителей, – способностью к адаптации путем изменения типа параллелизма [33] и числа параллельно функционирующих вычислительных структур, используемых при решении задач. Таким образом, ДРС в отличие от традиционных вычислителей обладают значительно большей гибкостью. Это, в частности, проявляется в том, что одна и та же ДРС в различные моменты времени может функционировать и как SISD-, и как MIMD-вычислитель [51].

В качестве примера практического применения рассматриваемой архитектуры можно привести устройства на базе отладочных модулей, изготавливаемых компаниями «Spectrum Digital» (США) и «Сигма- Интегрированные Системы» (Россия, г. Москва). Модули «TMS320DM642 Evaluation Module» (рис. 2.18) и «RMVideo-EVM» (рис. 2.19) включают ЦСП TMS320DM642, производимый компанией Texas Instruments Inc., и ДР ПЛИС различных фирм-производителей (в первом случае – ПЛИС семейства Spartan фирмы Xilinx, во втором случае – ПЛИС семейства Cyclone II фирмы Altera). Основные технические характеристики обоих модулей представлены соответственно в табл. 2.2 и 2.3 [45].

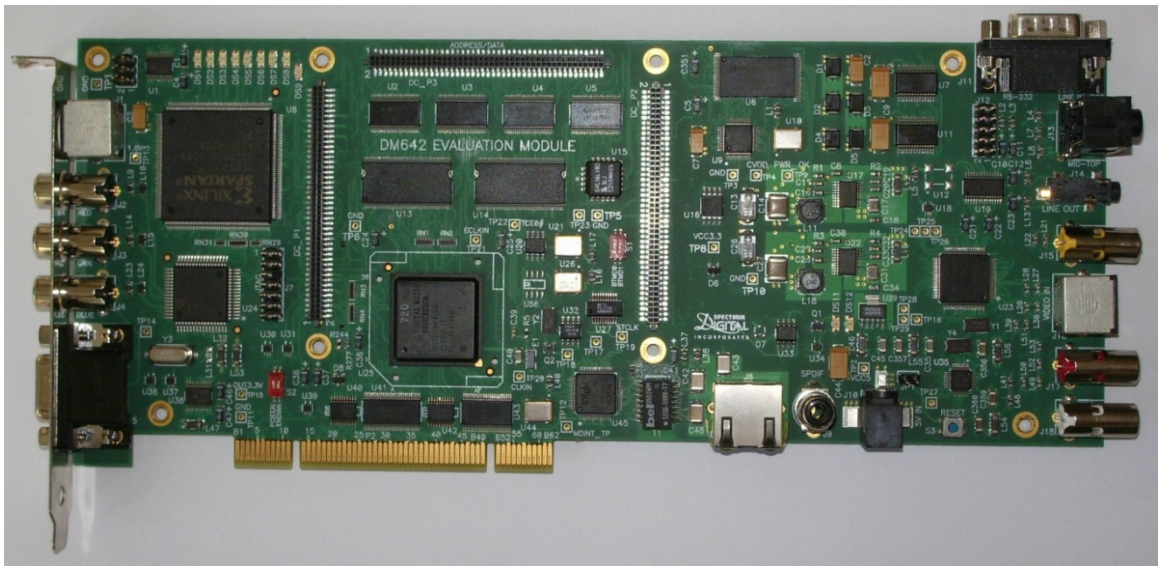


Рис. 2.18. Отладочный модуль «TMS320DM642 Evaluation Module»

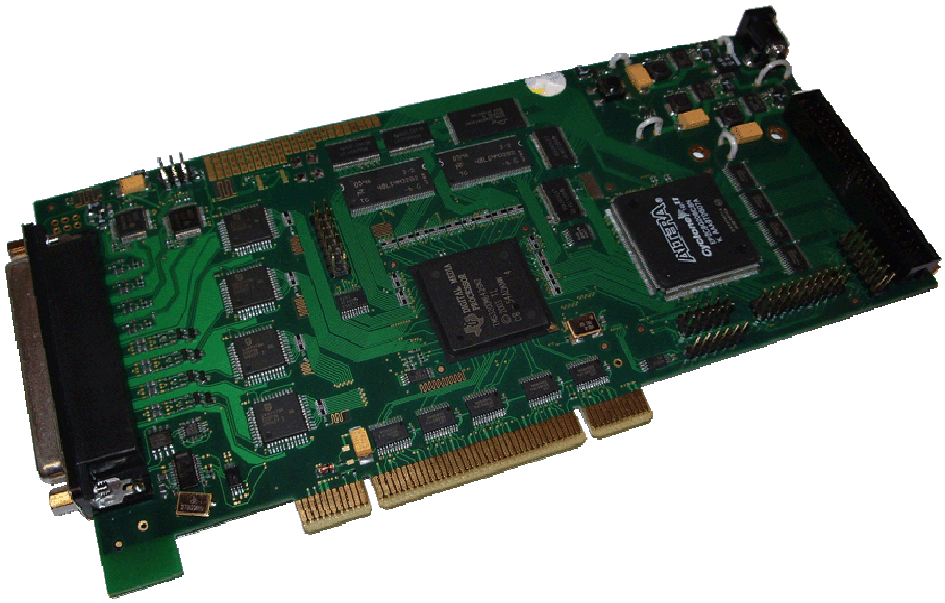


Рис. 2.19. Отладочный модуль «RMVideo-EVM»

Таблица 2.2

Технические характеристики отладочного модуля «TMS320DM642 Evaluation Module»

Элемент	Описание
Сигнальный процессор	TMS320DM642 (720 МГц)
ПЛИС (FPGA)	XC2S300 (Xilinx Spartan)
Динамическая ОЗУ с произвольным доступом (SDRAM)	MT48LC4M32B2 (4M×32×2)
Видео АЦП	TVP5416, TVP5150A
Видео ЦАП	SAA7105
Перепрограммируемая Flash-память	AM29LV033C (4 Мбайта)
Периферийные устройства ввода/вывода	PCI, Ethernet 10/100 Мбит

Таблица 2.3

Технические характеристики отладочного модуля«RMVideo-EVM»

Элемент	Описание
Сигнальный процессор	TMS320DM642 (720 МГц)
ПЛИС (FPGA)	EP2C8 (Altera Cyclone II)
Динамическая ОЗУ с произвольным доступом (SDRAM)	MT48LC4M32B2 (4M×32×2)
Видео АЦП	SAA7113×4
Видео ЦАП	SAA7120
Перепрограммируемая Flash-память	AT45DB081B (4 Мбайта)
Периферийные устройства ввода/вывода	PCI, Ethernet 10/100 Мбит

Как видно из табл. 2.2 и 2.3, на модулях отсутствуют RISC-процессоры, которые являются необходимым элементом предлагаемого решения. В этой связи был разработан внешний модуль (рис. 2.20), конструктивно имеющий такой же интерфейс, что и локальная шина PCI, и подключаемый таким же образом, как подключается шина PCI. На нем (с обратной стороны) размещен RISC-процессор MPC5121e с ядром ARM9, разработанный и изготавливаемый компанией Freescale (США). Именно данный модуль использовался для реализации устройства адаптивного сжатия данных, рассматриваемого в главе 4.

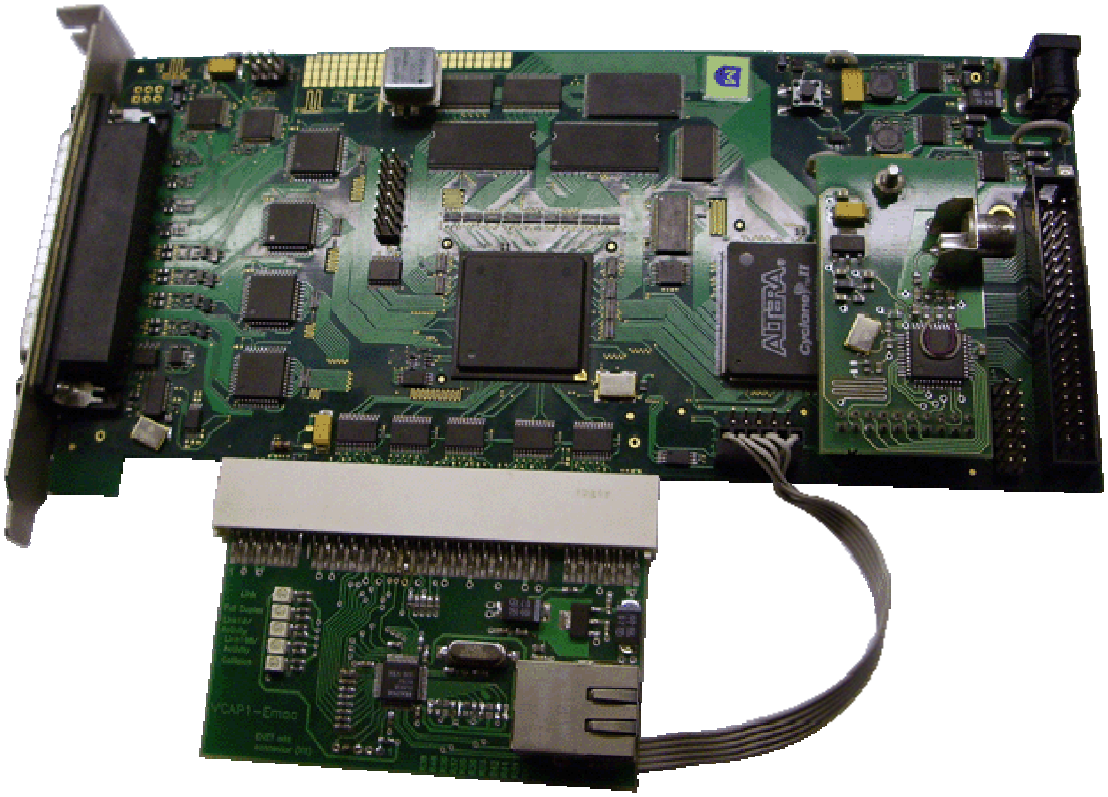


Рис. 2.20. Доработанный отладочный модуль «RMVideo-EVM» с RISC-процессором MPC5121e компании Freescale

Подводя итог проведенному обзору, можно сделать вывод, что к настоящему времени в развитии архитектурных решений для ДРС наметились две важные тенденции [51]:

− использование RISC-процессоров в качестве управляющего модуля ДРС;

− усложнение структуры ДРС.

Первая тенденция обусловлена тем фактом, что ДРС могут сочетать в себе преимущества CISC (Complex Instruction Set Computer – вычислитель с полным набором команд) и RISC (Reduced Instruction Set Computer – вычислитель с сокращенным набором команд) архитектур [43, 50]. В этом случае используются:

– ***RISC-процессор*** (например, на основе одного из семейств ARM, MIPS, SuperH, SPARC или другого), который применяется, в первую очередь, для выполнения операций управления и для взаимодействия с внешними устройствами;

– ***ДР ПЛИС***, на основе которой реализуется выполнение всех сложных математических операций и преобразований, минуя промежуточные пересылки данных, которые характерны для процессоров [7].

Для достижения наибольшей эффективности вышеописанной дуальной схемы (RISC-процессор и ДР ПЛИС) необходимо дополнить систему команд RISC-процессора специальными управляющими операциями, которые позволяют обеспечить наиболее эффективное взаимодействие с ДР ПЛИС (функциональное описание дополнительных операций и обоснование их целесообразности представлено в [33]). Следует отметить, что использование CISC-процессоров в составе подобных ДРС избыточно, так как сложные математические операции (в том числе и так называемые «составные операции» [7]) реализуются на базе ДР ПЛИС. Это связано с тем, что набор операций для ДРМ в отличие от традиционных процессоров не фиксирован. Таким образом, можно производить оптимизацию конфигурационных файлов для ДР ПЛИС под конкретную задачу, поскольку в этом случае отсутствует необходимость в универсальных решениях. При этом подобная ***ДРС будет оставаться универсальным вычислителем***.

Вторая тенденция является следствием естественного развития ДРС: в настоящее время наметился переход от базовых архитектурных решений [31, 33, 51, 53, 64, 71, 72], которые не учитывали многие особенности обработки информации на базе ДРС, к более сложным системам [43, 45, 10, 12, 13], лишенным подобных недостатков. Так, в [43, 44] были сформулированы основные принципы, позволяющие

соединить в одном вычислителе достоинства традиционных процессоров и ДРС. Следующим этапом стало создание гетерогенных (гибридных) вычислительных структур, объединяющих в себе различные типы процессоров и ДРС [10 − 13, 44], что сделало возможным использование преимуществ каждой из архитектур.

Кроме того, в качестве одного из основных направлений развития ДРС выделилось создание вычислительных комплексов, обладающих высокой производительностью при параллельной обработке данных. В этом случае ДРС может функционировать не только как SIMD-, но и как MIMD-вычислитель (Single Instruction Stream-Multiple Data Stream – одиночный поток команд и множественный поток данных, Multiple Instruction Stream-Multiple Data Stream – множественный поток команд и множественный поток данных), потому что ПЛИС допускает взаимодействие нескольких параллельно функционирующих процессов, которые обмениваются данными друг с другом. В этом случае выполнение одной команды над k потоками данных реализуется в виде k процессов в ДР ПЛИС. Причем значение переменной k может меняться от команды к команде, что делает подобные ДРС более гибкими, чем соответствующие мультипроцессорные MIMD-системы (например, CRAY Y-MP, Denelcor HEP, BBN Butterfly, Intel Paragon, CRAY T3D). Внутренняя структура ДР ПЛИС, функционирующей в режиме MIMD-вычислителя, представлена на рис. 2.21. Следовательно, классификация Флинна [63] в контексте ДРС должна трактоваться иначе, чем прежде, поскольку она применима не к ДРС в целом, а лишь к отдельным ее состояниям. Это еще раз подтверждает справедливость гипотезы о ДРС как об особом классе вычислителей, существенно отличающихся от фон-неймановских [33, 43, 50, 51].

Таким образом, число параллельно выполняемых команд и обрабатываемых потоков данных ограничивается не только логической емкостью ДР ПЛИС, их количеством в составе ДРС, но и пропускной способностью канала «память – ДР ПЛИС».

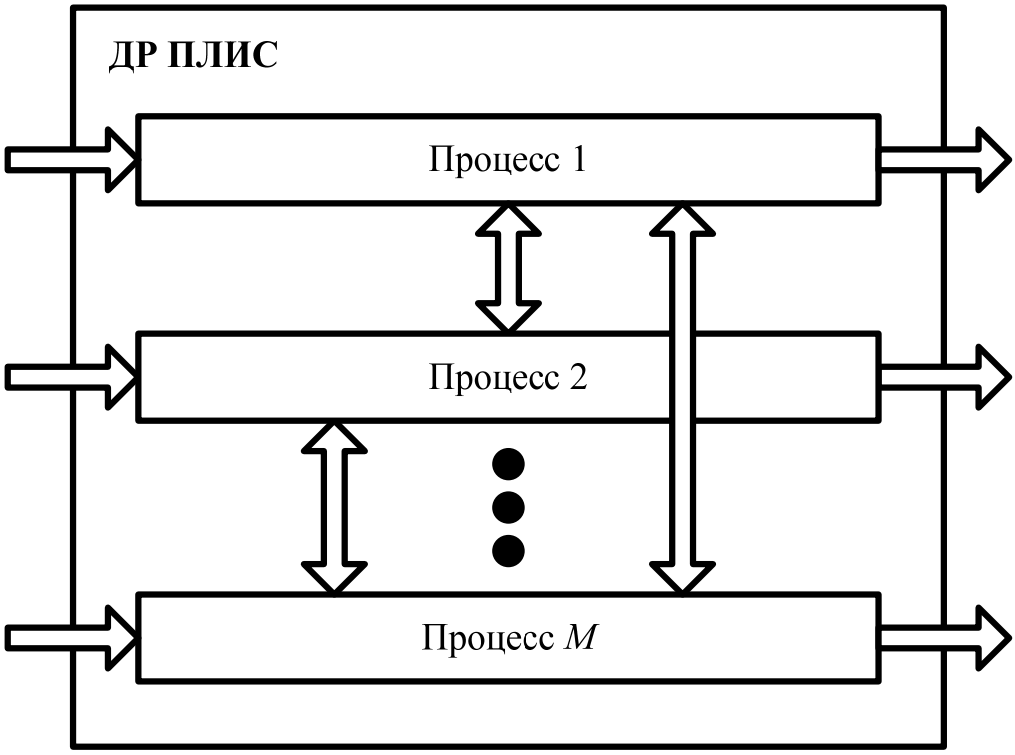


Рис. 2.21. Функционирование ДРС в режиме MIMD-вычислителя

ДРС позволяют по-новому взглянуть на способы реализации адаптивных систем: по определению СНАС не требуют изменения внутренней структуры, то есть использование ДРС для реализации СНАС не целесообразно. Для СОАС, напротив, применение ДРС является наиболее адекватным подходом, позволяющим обеспечить достаточную гибкость, сравнимую с программной реализацией, при значительно более высоком быстродействии. Следовательно, при реализации адаптивных систем на базе ДРС можно выделить два случая:

– ***системы без обратной связи:*** наиболее эффективной представляется реализация алгоритма адаптации на базе процессора (программно, что обусловлено его относительной простотой в случае отсутствия обратных связей), а УО – на базе ДР ПЛИС (аппаратно). Выбор описанного подхода обусловлен методом формирования алгоритма адаптации у СОАС без обратной связи – зависимость выявляется, формализуется и остается неизменной при обработке данных, что обуславливает относительную простоту расчетов, связанных с выбором соответствующих структур УО, а значит, и ДР ПЛИС. Пример реализации СОАС без обратной связи, предназначенной для вычисления элементарных и составных функций, рассмотрен в [32, 45, 48].

– ***системы с обратной связью:*** распределение функций между процессором и ПЛИС представляет собой отдельную научную задачу (в рамках направления co-design) – это относится как УО, так и к другим элементам адаптивной системы. Пример проектирования СОАС с обратной связью, предназначенной для компрессии видеоданных для узкополосных радиосистем, будет представлен в главе 4.

Особое значение имеет ДРС, впервые рассмотренная в [44]. Это обусловлено как новизной архитектурных решений, так и важностью области ее применения. Предложенный подход к проектированию высоконадежных вычислительных систем открывает новую нишу в применении ДРС.

Как известно, самым важным требованием, предъявляемым к электронной аппаратуре, применяемой в системах специального назначения (например, в авиационных бортовых системах, в системах управления атомными станциями и в космических летательных аппаратах), является высокая надежность ее функционирования, что обеспечивается на этапе проектирования, производства и эксплуатации. На стадии проектирования этот показатель качества определяется, прежде всего, алгоритмическим и аппаратно-программным обеспечением. Исследования в области надежности алгоритмического обеспечения относятся, прежде всего, к вопросам, изучаемым прикладной математикой. Однако применение даже качественных алгоритмических решений само по себе не может обеспечить высокой надежности системы целиком, пока не гарантирована работоспособность аппаратного и программного обеспечения. Именно поэтому в настоящей работе будут рассмотрены вопросы, связанные с обеспечение высокой надежности аппаратных средств и системного программного обеспечения [44].

Как известно, существуют различные методы повышения надежности [55]. Методы, используемые для повышения надежности на этапе проектирования, могут быть классифицированы, например, следующим образом [44]:

1) ***физические методы*** связаны с улучшением качества изделий за счет совершенствования технологии их изготовления, применения более качественных материалов и т.п. (например, на печатных платах

целесообразно минимизировать число паяных соединений, обладающих крайне низкой надежностью, путем перенесения этих соединений непосредственно на кристалл за счет применения микросхем с высокой степенью интеграции);

2) ***электрические методы*** позволяют повысить надежность изделий за счет снижения электрических нагрузок и устранения режима перегрузок (в нормальном рабочем режиме) как на все изделие, так и на его отдельные компоненты (например работа процессора в режиме со сниженной тактовой частотой); очевидно, что главный недостаток такого подхода – увеличение себестоимости изделия;

3) ***структурно-схемотехнические методы*** заключаются в применении таких структурных и схемотехнических решений, которые позволяют существенно снизить вероятность отказа устройства в целом (наиболее известный метод − резервирование, то есть кратное увеличение числа элементов, способных выполнять одинаковые функции). К недостаткам резервирования можно отнести следующее:

а) увеличение стоимости прямо пропорционально кратности резервирования;

б) увеличение числа паяных соединений на печатной плате;

в) если при резервировании используется одна и та же элементная база, которая может иметь одинаковые физические недостатки, то это существенно снижает эффективность подобного подхода.

Следует подчеркнуть, что обеспечение надежности состоит не только в защите от внешних, но и от внутренних дестабилизирующих воздействий. К внутренним дестабилизирующим воздействиям относятся ошибки в аппаратном и программном обеспечении, которые не были устранены на этапе проектирования, не были обнаружены при испытаниях или были привнесены на этапе производства электронных средств. Наличие ошибок в аппаратном и программном обеспечении является вполне объективным фактором, который принципиально нельзя исключать из рассмотрения в виду высокой сложности современных электронно-вычислительных устройств [44].

В случае ответственных применений можно комбинировать сразу все три указанных подхода, применяя в вычислителях различные

СБИС, способные выполнять одинаковые функции (данное свойство элементной базы назовем «***функциональной альтернативностью***»). Таким образом, выход из строя одного вычислительного узла приводит к тому, что его задачи перераспределяются между другими. При этом целесообразным представляется следующая функция: при нормальной работе вычислительные узлы не просто дублируют друг друга, увеличивая общее энергопотребление системы, а обеспечивают расширенную функциональность. По мере отказа отдельных узлов происходит реконфигурация системы с целью обеспечения возможности дальнейшей ее работы – при этом постепенно сокращаются ее функциональные возможности. Кроме того, поскольку элементная база выбирается с таким расчетом, что вычислительных ресурсов больше, чем требуется в нормальном рабочем режиме для выполнения всех поставленных задач, то можно существенно снизить нагрузки на основные вычислительные узлы. По мере отказа отдельных узлов нагрузка на работающие элементы электронно-вычислительной подсистемы возрастает. Следует заметить, что отказом узла считается не только отказ самого вычислительного элемента (например, процессора или ПЛИС), но и периферийных элементов, без которых дальнейшая работа узла не представляется возможной (например, оперативной памяти или контроллера интерфейса, по которому происходит взаимодействие с другими узлами) [44].

В качестве элементной базы, обладающей свойством функциональной альтернативности, могут выступать программируемые логические интегральные схемы и цифровые сигнальные процессоры. Применение этой элементной базы обусловлено целым рядом факторов:

− и ПЛИС, и ЦСП способны вполне успешно решать задачи цифровой обработки сигналов;

− как правило, компании, специализирующиеся на разработке и производстве ЦСП, не выпускают ПЛИС, и наоборот, что обуславливает существенные различия в формируемых полупроводниковых структурах (как за счет использования принципиально отличающихся схемотехнических решений, так и различных технологических процессов) [44].

В системах с высокой надежностью целесообразно применение ДР ПЛИС, что обусловлено целым рядом факторов. Во-первых, они способны менять свою внутреннюю логическую структуру непосредственно в процессе работы, что является крайне полезным свойством при перераспределении ресурсов в случае отказа части вычислительных узлов. Во-вторых, использование ДР ПЛИС в тандеме с процессором (процессорами) позволяет существенно повысить производительность электронно-вычислительной подсистемы [10 − 13, 45, 49, 59 − 61], в том числе при реализации адаптивных алгоритмов цифровой обработки сигналов [45]. Таким образом, задача построения высоконадежной электронной аппаратуры для ответственных применений может быть решена с использованием динамически реконфигурируемых систем обработки информации [44].

Пример архитектуры подобной системы, построенной на основе ДРС, представлен на рис. 2.22. Данный вычислитель состоит из трех основных вычислительных модулей: ЦСП, ДР ПЛИС и RISC-процессора.

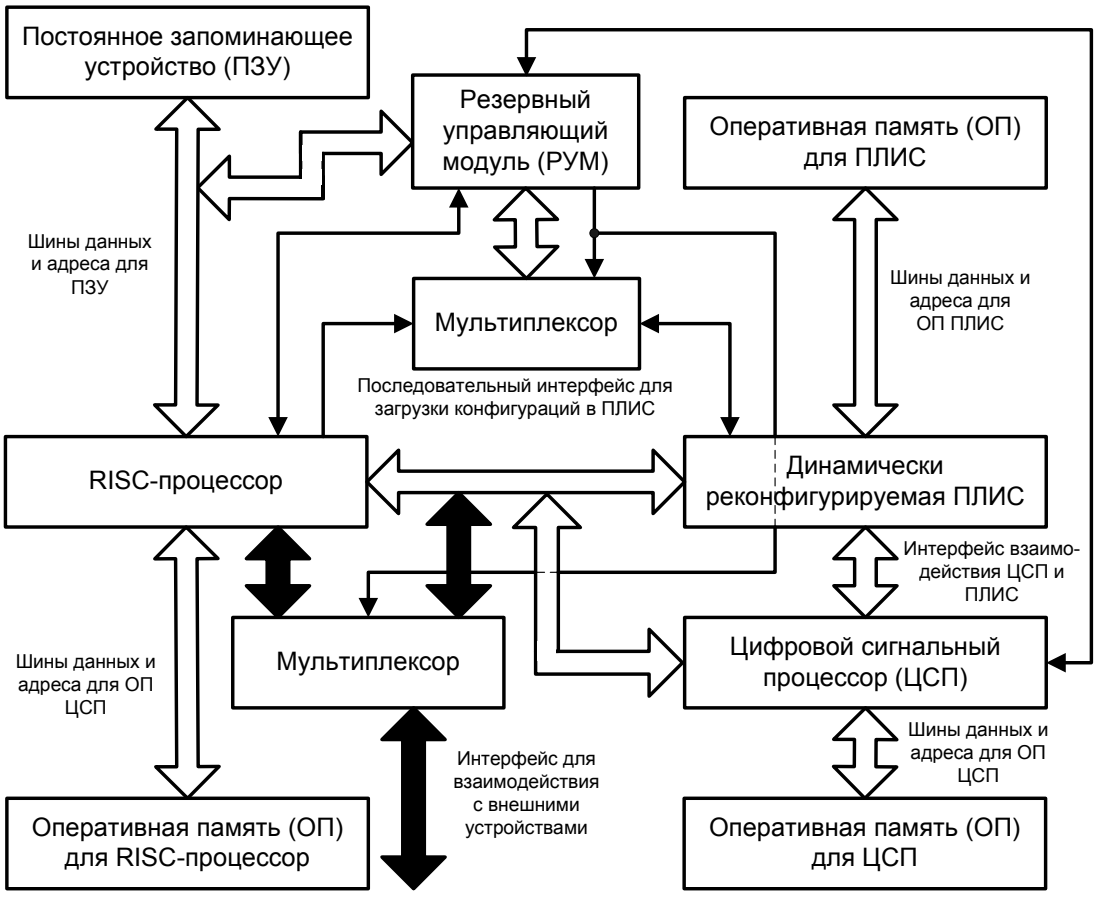


Рис. 2.22. Обобщенная структурная схема высоконадежного динамически реконфигурируемого вычислителя

Как уже было сказано выше, основная функция ЦСП и ДР ПЛИС – решение задач цифровой обработки сигналов, а также выполнение других операций, обладающих большой вычислительной сложностью. Для RISC-процессора (Reduced Instruction Set Computer − вычислитель с сокращенной системой команд), напротив, основная цель в нормальном режиме работы заключается в управлении системой, диспетчеризации задач между основными вычислительными узлами, во взаимодействии с внешними устройствами и пользователем. В случае отказа ДР ПЛИС и/или ЦСП вычислительная нагрузка на RISC-процессор возрастает [44].

К каждому вычислительному модулю подключена собственная оперативная память (ОП), поэтому обмен данными между устройствами происходит путем их передачи или по высокоскоростной шине, соединяющей все три модуля, или по специальному параллельному интерфейсу между ЦСП и ДР ПЛИС. Кроме того, для загрузки конфигурационного файла в ДР ПЛИС используется последовательный интерфейс (например, JTAG) между RISC-процессором и ДР ПЛИС. Для хранения конфигурационных файлов используется постоянное запоминающее устройство (ПЗУ), построенное по твердотельной технологии (например, однократно программируемая память – One-Time-Programmable Memory или flash-память) [44].

Каждый вычислительный модуль выполняет проверку целостности данных, хранящихся в подключенной к нему ОП. Проверку целостности данных, хранящихся в ПЗУ, в нормальном режиме работы осуществляет RISC-процессор. В случае выхода его отказа эти функции берет на себя резервный управляющий модуль (РУМ, Standby Control Unit, SCU). При отказе последнего функции проверки целостности распределяются между функционирующими вычислительными модулями (ЦСП и/или ДР ПЛИС) [44].

РУМ является резервным модулем, который необходим для выполнения части функций RISC-процессора в случае отказа последнего и который может быть выполнен как на базе ПЛИС сравнительно небольшой логической емкости, так и на базе относительно простого микроконтроллера (например, 16- или 32-разрядного) [44]. В обязательном порядке РУМ реализует следующие функции:

− сторожевого таймера,

− проверки целостности данных,

− управления мультиплексорами для изменения направлений потоков данных.

Кроме того, предлагаемый динамически реконфигурируемый вычислитель включает два мультиплексора, предназначенных для переключения потоков данных между RISC-процессором и РУМ (см. рис. 2.22). Следует подчеркнуть, что в рассматриваемом вычислителе по сути реализована схема резервирования с параллельным соединением резервирующих элементов. Для того чтобы наглядно обосновать превосходство предлагаемого подхода над традиционным вариантом, рассмотрим схему с параллельным соединением резервирующих элементов (рис. 2.23) и сравним вероятности отказов при реализации классического и усовершенствованного подходов [44].

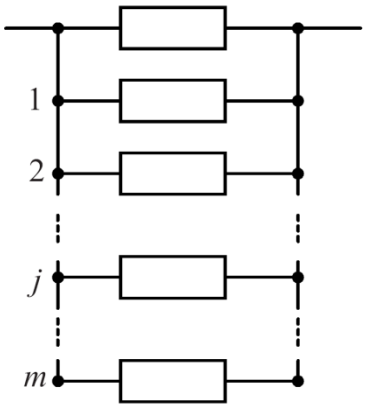


Рис. 2.23. Схема параллельного соединения резервирующих элементов

Пусть событие F_j состоит в отказе *j*-го элемента, вероятность этого события равна $p_t(F_j)$. Индекс *t* указывает на то, что вероятность отказа *j*-го элемента является функцией времени. Тогда вероятность отказа всей системы, состоящей из $(m+1)$-го элемента, состоит в одновременном возникновении событий $F_0F_1F_2...F_j...F_{m-2}F_{m-1}F_m$ и вычисляется по формуле [16]:

$$P_t\left(\prod_{j=0}^{m} F_j\right) = \prod_{j=0}^{m} p_t(F_j). \qquad (2.1)$$

Однако это справедливо только для независимых случайных величин. Очевидно, что для системы с резервированием, построенной на основе одной и той же элементной базы, такие события нельзя считать независимыми. Тогда в соответствии с теоремой умножения вероятностей, вероятность одновременного возникновения событий $F_0, F_1, F_2, ..., F_j, ..., F_{m-2}, F_{m-1}, F_m$ вычисляется по формуле [16]:

$$P_t\left(\prod_{j=0}^{m} F_j\right) = p_t(F_0)\, p_t(F_1 \mid F_0)\, p_t(F_2 \mid F_0 F_1) \ldots p_t(F_m \mid F_0 F_1 \ldots F_{m-1}). \quad (2.2)$$

В случае применения различной элементной базы и при выполнении всех рекомендаций по проектированию электронной аппаратуры для специальных применений, события $F_0, F_1, F_2, ..., F_j, ..., F_{m-2}, F_{m-1}, F_m$ можно считать квазинезависимыми и вероятность отказа всей системы рассчитывать с использованием формулы (2.1) [44].

Вероятность отказа предлагаемого вычислителя отличается для различных режимов его работы. Чтобы оценить динамику вероятности отказа, следует рассмотреть граф состояний высоконадежного динамически реконфигурируемого вычислителя (рис. 2.24, табл. 2.4).

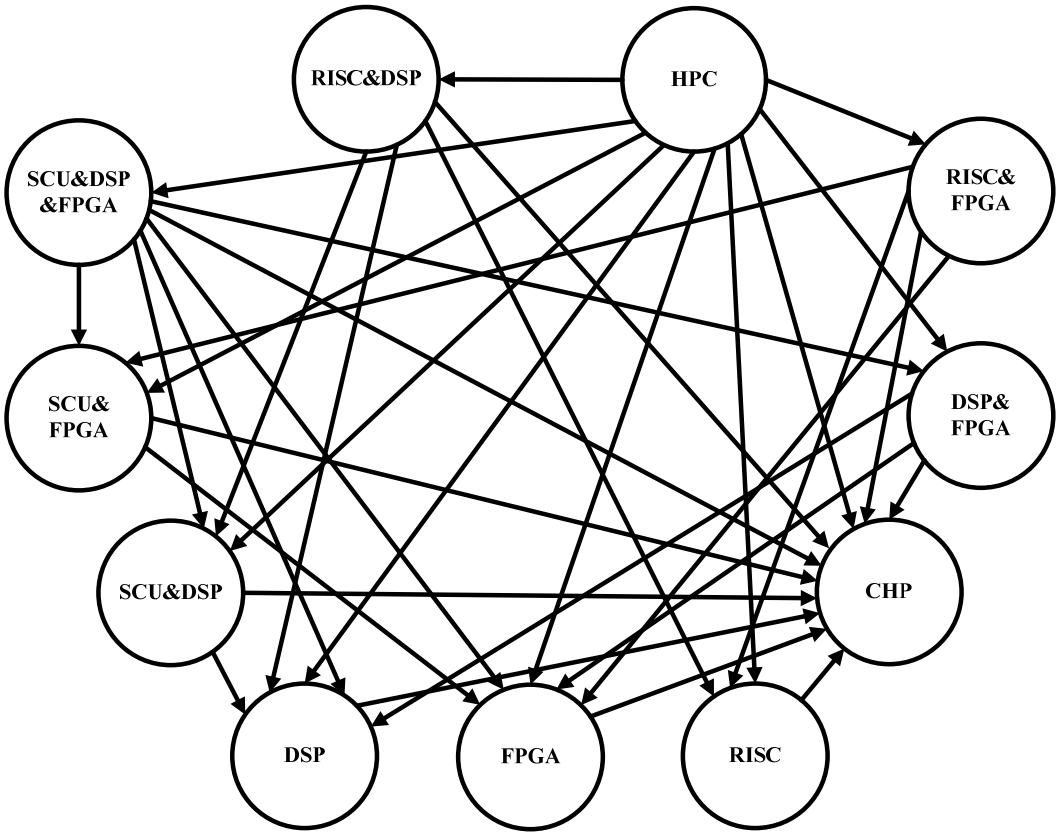


Рис. 2.24. Граф состояний высоконадежного динамически реконфигурируемого вычислителя

Таблица 2.4

Оценки вероятностей изменения состояний высоконадежного вычислителя

Состояние		Вероятность изменения состояния
Исходное	Конечное	
HPC	Состояние работоспособности РУМ, ЦСП и ДР ПЛИС (SCU&DSP&FPGA)	p_{RISC}
HPC	Состояние работоспособности RISC-процессора и ЦСП (RISC&DSP)	p_{FPGA}
HPC	Состояние работоспособности RISC-процессора и ДР ПЛИС (RISC&FPGA)	p_{DSP}
HPC	Состояние работоспособности РУМ и ЦСП (SCU&DSP)	$p_{RISC}p_{FPGA}$
HPC	Состояние работоспособности РУМ и ДР ПЛИС (SCU&FPGA)	$p_{RISC}p_{DSP}$
HPC	Состояние работоспособности ЦСП и ДР ПЛИС (DSP&FPGA)	$p_{RISC}p_{SCU}$, где p_{SCU} – вероятность отказа РУМ
HPC	Состояние работоспособности RISC-процессора (RISC)	$p_{DSP}p_{FPGA}$
HPC	Состояние работоспособности ЦСП (DSP)	$p_{RISC}p_{SCU}p_{FPGA}$
HPC	Состояние работоспособности ДР ПЛИС (FPGA)	$p_{RISC}p_{SCU}p_{DSP}$
HPC	Состояние неработоспособности (CHP)	$p_{RISC}p_{DSP}p_{FPGA}$
SCU&DSP&FPGA	SCU&DSP	p_{FPGA}
SCU&DSP&FPGA	SCU&FPGA	p_{DSP}
SCU&DSP&FPGA	DSP&FPGA	p_{SCU}
SCU&DSP&FPGA	DSP	$p_{SCU}p_{FPGA}$
SCU&DSP&FPGA	FPGA	$p_{SCU}p_{DSP}$
SCU&DSP&FPGA	CHP	$p_{SCU}p_{DSP}p_{FPGA}$
RISC&DSP	RISC	p_{DSP}

Окончание табл. 2.4

Состояние		Вероятность изменения состояния
Исходное	Конечное	
RISC&DSP	SCU&DSP	p_{RISC}
RISC&DSP	DSP	$p_{RISC}p_{SCU}$
RISC&DSP	CHP	$p_{RISC}p_{DSP}$
RISC&FPGA	RISC	p_{FPGA}
RISC&FPGA	SCU&FPGA	p_{RISC}
RISC&FPGA	FPGA	$p_{RISC}p_{SCU}$
RISC&FPGA	CHP	$p_{RISC}p_{FPGA}$
SCU&DSP	DSP	p_{SCU}
SCU&DSP	CHP	$p_{SCU}p_{DSP}$
SCU&FPGA	FPGA	p_{SCU}
SCU&FPGA	CHP	$p_{SCU}p_{FPGA}$
DSP&FPGA	DSP	p_{FPGA}
DSP&FPGA	FPGA	p_{DSP}
DSP&FPGA	CHP	$p_{DSP}p_{FPGA}$
RISC	CHP	p_{RISC}
DSP	CHP	p_{DSP}
FPGA	CHP	p_{FPGA}

Основным режимом работы устройства является так называемое «нормальное рабочее состояние» (НРС), то есть состояние, в котором все основные вычислительные модули и все периферийные устройства работоспособны. Для этого состояния вероятность отказа всей системы, то есть вероятность перехода в состояние неработоспособности (СНР), P_{Sys} минимальна:

$$P_{Sys} = p_{RISC}p_{DSP}p_{FPGA}, \tag{2.3}$$

где p_{RISC} – вероятность отказа RISC-процессора; p_{DSP} – вероятность отказа ЦСП; p_{FPGA} – вероятность отказа ДР ПЛИС [44].

Остальные состояния обозначены через названия работоспособных вычислительных модулей. Оценки вероятности перехода из од-

ного состояния в другое представлены в табл. 2.4. Вероятность отказа всей системы, совершившей n переходов из одного состояния в другое, P_{Sys} состоит в последовательном возникновении событий $S_0 \to S_1$, $S_1 \to S_2$, $S_2 \to S_3$, …, $S_{i-1} \to S_i$, …, $S_i \to S_{i+1}$, …, $S_{n-2} \to S_{n-1}$, $S_{n-1} \to S_n$ и вычисляется по формуле [44]

$$P_{Sys} = \prod_{i=0}^{n-1} p(S_i \to S_{i+1}). \qquad (2.4)$$

Очевидно, что состояние S_0 соответствует НРС, S_n – СНР, а максимальное значение числа переходов для рассматриваемой архитектуры $n_{\max} = 4$. Таким образом, используя формулу (2.4) и данные табл. 2.4, можно рассчитать значения вероятности каждого из вариантов отказа предлагаемого динамически реконфигурируемого вычислителя [44].

При оценке вероятности отказа не учитывается вероятность отказа ПЗУ, которую обозначим через p_{ROM}. Это связано с тем, что устройство не становится неработоспособным, поскольку все необходимые для работы в текущем режиме данные должны быть заранее загружены в ОЗУ каждого вычислительного модуля. Однако дальнейшие реконфигурации системы становятся невозможными, так как конфигурационные файлы для каждого из вычислительных модулей будут недоступны. В этом случае отказ любого модуля в любом из состояний ведет к отказу системы в целом. По этой причине целесообразно выбирать высоконадежные ПЗУ для применения в подобных системах (в частности, можно рекомендовать резервирование ПЗУ с использованием памяти различных типов и различных фирм-производителей) [44].

Кроме того, следует отметить, что данный граф может описывать состояния не только аппаратного, но и программного обеспечения. В последнем случае переходы будут двунаправленными, поскольку существует возможность восстановления работоспособности модуля после его перезагрузки [44].

При выборе элементной базы для предлагаемого динамически реконфигурируемого вычислителя следует обращать внимание на компании-производители ЦСП и RISC-процессоров. Рекомендуется в

рамках одного вычислителя комбинировать различные марки (например, RISC-процессоры на основе ARM9-ядра от Atmel, ЦСП семейства C6000 от Texas Instruments, ДР ПЛИС от Altera, альтернативный вариант − RISC-процессоры на основе ядра PowerPC от Freescale, ЦСП семейства TigerShark от Analog Devices, ДР ПЛИС от Xilinx), чтобы снизить вероятность отказа вычислителя целиком, избегая одинаковых ошибок, допущенных сотрудниками компании как на этапе проектирования, так и производства. Число подобных вычислителей, дублирующих друг друга, может изменяться в зависимости от решаемых задач [44].

Предложенный подход позволяет снизить вероятность отказов вычислителя при возникновении ошибок в системном программном обеспечении. Рассмотрим это подробнее на примере операционных систем реального времени (ОСРВ; Real-Time Operating Systems (RTOS)). Они являются единственно возможным вариантом для ответственных применений, в том числе и для систем специального назначения. В качестве ОСРВ, которые способны функционировать на различных платформах, могут быть использованы следующие: QNX (QNX Software Systems, Канада), LynxOS (LynuxWorks, США), VxWorks (Wind River Systems, США). Следует заметить, что ОСРВ QNX версии 4.25 (изделие КПДА.00001-01) [40] и защищенная ОСРВ «QNX» (изделие КПДА.00002-01) [35] сертифицированы Федеральной службой по техническому и экспортному контролю Российской Федерации. Кроме того, можно использовать специализированные ОСРВ, разрабатываемые фирмами-производителями элементной базы. В качестве примера такой операционной системы можно привести ОСРВ DSP/BIOS, разработанную компанией Texas Instruments для применения в своих цифровых сигнальных процессорах. Для ДР ПЛИС можно использовать самые различные операционные системы, если внутри микросхемы будет реализовано процессорное ядро. Во многих случаях для повышения надежности функционирования системы от использования процессорных ядер внутри ДР ПЛИС и, следовательно, операционных систем можно вообще отказаться. Это обусловлено той ролью, которую играет ДР ПЛИС в системе. Как из-

вестно, главным недостатком вычислительных устройств, где отсутствует операционная система, является время реализации необходимых алгоритмов. Однако ДР ПЛИС предназначена, в первую очередь, для реализации вычислений специальных функций и выполнения расчетов, обладающих значительным параллелизмом. В нормальном рабочем режиме на нее не ложатся задачи общесистемного управления. Только в случае отказа обоих процессоров ДР ПЛИС должна будет заниматься управлением всей системой, но для этого ей потребуется выполнять только базовые функции, поскольку функциональность в данном режиме сильно ограничена [44].

Таким образом, предлагаемый подход позволяет комплексно решать проблему обеспечения надежности аппаратного и программного обеспечения, допуская возможность одновременного воздействия множества внешних (например, воздействие ионизирующего излучения и механических воздействий) и внутренних дестабилизирующих факторов [44].

2.4. Классификация динамически реконфигурируемых систем обработки информации

Как показала практика, классификация вычислительных устройств – это нечто большее, чем просто результат анализа и систематизации известных научных результатов (достаточно вспомнить таксономию Флинна, оказавшую огромное влияние на все последующее развитие вычислительной техники). В этой связи для такого относительно нового направления, каким являются ДРС, разработка различных критериев и схем классификации имеет особое значение, позволяя определить основные направления их развития.

Сами классификации можно разделить на две большие группы:

− определяющие особенности ДРС как целостной вычислительной системы;

− характеризующие различные особенности построения и функционирования отдельных элементов ДРС, в частности, ДРМ или управляющего модуля (контроллера) ДРС.

К первой группе относится, прежде всего, классификация, разделяющая ДРС по принципу наличия в ее составе полнофункционального процессора. В этом случае можно выделить две группы динамически реконфигурируемых систем обработки информации:

1) ***простые*** – это такие ДРС, у которых управляющий модуль не обладает функционально полной системой команд, то есть контроллер ДРС при отсутствии ДРМ не удовлетворяет принципу Чёрча-Тьюринга для универсальных вычислителей (к этой группе принадлежит АВУ);

2) ***гибридные*** – это такие ДРС, в состав которых входит полнофункциональный процессор, который способен решить любую алгоритмически разрешимую задачу (в качестве примера такой ДРС можно привести КВУ, ДРВ, DISC и другие).

В настоящее время известно достаточно большое число классификаций, касающихся отдельных элементов ДРС. В настоящем разделе будут приведены только наиболее распространенные. Первой и самой очевидной классификацией является разделение ДРС по числу входящих в ее состав вычислительных модулей (в том числе ДРМ). Обозначим число вычислительных узлов в составе ДРС через N_{CM}, а число ДРМ – через N_{DRM}. Очевидно, что для любой ДРС всегда справедливо неравенство:

$$N_{CM} \geq N_{DRM}. \qquad (2.5)$$

Применяя эту классификацию к АВУ, DISC, ДРС, реализованным в PCI-модулях семейства WILDFORCE, и ДРС для решения задач цифровой обработки сигналов, можно утверждать, что:

− АВУ – это ДРС, имеющая в своем составе один вычислительный узел, которым является ДРМ, то есть $N_{CM} = N_{DRM} = 1$;

− DISC – это ДРС, для которой $N_{CM} = 2$ и $N_{DRM} = 1$;

− для ДРС, использованных в PCI-модулях семейства WILDFORCE, $N_{CM} = 6$ и $N_{DRM} = 5$;

− у ДРС для решения задач цифровой обработки сигналов $N_{CM} = 3$ и $N_{DRM} = 1$.

Другая классификация разделяет ДРС по функциям, возложенным на управляющий модуль и ДРМ. В соответствие с этим принципом можно выделить ДРС, у которых:

− ***ДРМ*** выполняет роль ***арифметико-логического устройства*** (АЛУ), ***сопроцессора***, ***препроцессора*** или ***единого пре- и сопроцессора***;

− ***управляющий модуль*** (контроллер) ДРС занимается исключительно ***задачами управления*** (в том числе диспетчеризацией и является арбитром при доступе к общим для вычислительных узлов ресурсам) или совмещает в себе функции ***управляющего и вычислительного узлов***.

Следует подчеркнуть, что у ДРС функции как управляющего, так и динамически реконфигурируемого модулей могут меняться с течением времени, поэтому данная классификация отражает только текущее состояние системы.

Следующая классификация отражает участие ДРМ (или даже ДРС целиком) в распределении ресурсов вычислительного комплекса между задачами. Она может быть применима не только к ДРС, но и к любым другим вычислительным устройствам. При этом данная классификация связана не только с архитектурой ДРС, но и с архитектурой надсистемы, в состав которой входит рассматриваемая ДРС. В соответствии с данным критерием можно выделить ***две группы вычислительных узлов***:

− основные, то есть такие узлы, которые участвуют в процессе распределения между задачами вычислительных ресурсов всей системы;

− вспомогательные, то есть те узлы, которые в процессе распределения вычислительных ресурсов не участвуют.

В качестве примера можно рассмотреть систему SPLASH-II, показанную на рис. 2.11: основным вычислительным узлом системы в целом является компьютер SUN SPARC, поскольку именно он занимается распределением ресурсов системы, а ДРС SPLASH-II выступает в качестве вспомогательного узла. Другой вариант представлен на рис. 2.8: в лабораторном стенде реализована ДРС с архитектурой ДРВ. В этом случае ДРС и является основным вычислительным узлом. Сам ДРВ, как уже упоминалось выше, включает в себя микро-

контроллер (управляющий модуль) и ДР ПЛИС (ДРМ). В случае ДРВ микроконтроллер является основным вычислительным узлом в составе ДРС, а ДР ПЛИС – вспомогательным. Дополнительные пояснения по данной классификации приведены в приложении А.

Последняя классификация отражает то, каким образом взаимодействует ДРМ с другими модулями, входящими в состав ДРС. Применяя данный критерий, можно выделить следующие типы ДРМ:

− ***жестко связанный*** – это такой ДРМ, который связан напрямую только с одним управляющим модулем и который не может быть использован без обращения к указанному управляющему модулю (к таким ДРМ, в том числе, относятся динамически реконфигурируемые пре- и сопроцессоры);

− ***свободно связанный*** (loosely coupled) – это такой ДРМ, который может взаимодействовать (одновременно или поочередно) с несколькими (как с управляющими, так и с другими динамически реконфигурируемыми) модулями. Примерами свободно связанных ДРМ могут служить такие вычислительные узлы, у которых управляющим модулем является центральный процессор компьютера и которые подключаются к последнему через локальную или системную шину.

Рассмотренный критерий может быть применен не только к отдельным модулям, но и к ДРС в целом.

Завершая данный раздел, следует подчеркнуть, что ДРС – это достаточно новая область вычислительной техники, поэтому предложенные подходы к классификации рассматриваемых систем могут быть в будущем скорректированы и дополнены. Однако основные критерии, лежащие в основе описанных классификаций, получены из опыта практического применения ДРС и теоретически обоснованы, а это позволяет считать, что рассмотренные принципы будут применяться и в дальнейшем.

Вопросы и задания и для самоподготовки

1. Что такое адаптивное вычислительное устройство? Опишите его основополагающие принципы и работу. Чем АВУ отличается от традиционных процессоров, построенных на принципах фон Неймана? Что понимается под гарвардской архитектурой АВУ?

2. Что представляет собой комплементарное вычислительное устройство? Чем оно отличается от АВУ? В каких случаях обоснованно использование КВУ в качестве альтернативы АВУ? Приведите примеры конкретных задач, где целесообразно применять КВУ.

3. Опишите архитектуру ДРС типа «динамически реконфигурируемый вычислитель» (ДРВ). Какова, на Ваш взгляд, область его применений? Какими ограничениями обладает данная архитектура?

4. Что такое «вычислитель с динамически изменяемой системой команд» (DISC)? Сравните ДРС типа ДРВ и «вычислитель с динамически изменяемой системой команд». Чем отличаются области применения данных архитектур? Какими архитектурными особенностями обуславливаются отличия между ДРВ и DISC?

5. Что представляет собой систолическая архитектура вычислителей? В каких случаях целесообразно применять ее в ДРС? Почему? Приведите примеры возможных приложений вычислителей с систолической архитектурой.

6. Опишите ДРС с систолической архитектурой SPLASH-II. В чем состоит ее специфика?

7. Каким образом происходит обмен данными между систолами и центральным вычислителем в ДРС, построенных на базе архитектуры SPLASH-II?

8. Дайте краткую характеристику PCI-модулям семейства WILDFORC. Какова область применения подобных модулей? Сравните ДРС с архитектурой SPLASH-II и PCI-модулями семейства WILDFORC. Чем отличается область применения каждого варианта?

9. Опишите архитектуру PAM. Чем она отличается от систолических архитектур, подобных SPLASH-II? В чем заключаются преимущества и недостатки архитектуры типа PAM?

10. Что такое гетерогенные ДРС? Опишите, каким образом функционируют гетерогенные ДРС. Приведите примеры гетерогенных ДРС.

11. Дайте характеристику гетерогенной ДРС, ориентированной на задачи цифровой обработки сигналов. Нарисуйте структурную схему указанной ДРС. Какие элементы входят в состав данной ДРС? Как

они взаимодействуют между собой? Перечислите основные технические характеристики электронно-вычислительных устройств, в которых реализована подобная архитектура.

12. Приведите примеры, когда требуется функционирование ДРС в режиме MIMD-вычислителя. Каким образом может быть реализован подобный режим?

13. Опишите, каким образом реализуются адаптивные системы на базе ДРС. Чем отличается реализация адаптивных систем с обратной связью и без неё?

14. Каким образом обеспечивается надежность электронно-вычислительных средств на этапе их проектирования? Как архитектура вычислителя может повлиять на его надежность?

15. Нарисуйте структурную схему ДРС для ответственных применений. Какие элементы входят в ее состав? Как между ними распределяются задачи, решаемые вычислителем? Как они взаимодействуют друг с другом?

16. Что происходит, если отказывает один или несколько элементов ДРС для ответственных применений? Приведите формализованные оценки надежности подобного вычислителя. Какое влияние оказывает выбор операционных систем на надежность ДРС в целом?

17. Какие архитектуры ДРС, не описанные выше, Вам известны? В чем состоит отличие приведенных Вами архитектур от рассмотренных?

18. Используя доступные источники информации, проведите сравнительный анализ архитектур ДРС, не рассмотренных в настоящем учебном пособии. Результаты анализа сведите таблицу, которую представите преподавателю в качестве отчета.

19. Как конструктивно реализуются ДРС? Приведите примеры. Поясните, почему именно такие конструкторские решения были использованы.

20. Какие классификации ДРС Вам известны? Чем они отличаются друг от друга? Что они характеризуют? Какие классификации ДРС могли бы предложить Вы?

ГЛАВА 3. ОСНОВЫ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИ РЕКОНФИГУРИРУЕМЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ

Основная цель математического моделирования ДРС − определение времени выполнения той или иной вычислительной задачи в зависимости от особенностей используемый архитектуры. Таким образом, математическое моделирование – один из ключевых аспектов проектирования подобных вычислительных устройств, поскольку позволяет на самых ранних этапах проектирования оценить, удовлетворяет ли разрабатываемая система требованиям технического задания или нет; в процесс отладки встраиваемого программного обеспечения (embedded software) рассчитывать общее время выполнения решаемой задачи, а не отдельных процедур, выполняемых на базе управляющего модуля и ДРМ.

Таким образом, наличие математической модели ДРС − необходимая предпосылка для создания единого компилятора и единой среды программирования, которая будет включать в качестве отдельных программных модулей систему автоматизированного проектирования ДРМ и среду программирования управляющего модуля (например, на базе RISC-процессора).

3.1. Общие положения, касающиеся математического моделирования динамически реконфигурируемых систем обработки информации

Очевидно, что при разработке математической модели ДРС нельзя ограничиваться частными случаями, которые дают хорошие результаты для конкретной архитектуры и известной элементной базы, по двум причинам. Во-первых, число новых архитектурных решений для ДРС растет чрезвычайно быстро. В этой связи создание множества разрозненных математических моделей малоэффективно. Во-вторых, уже в настоящее время существуют варианты, альтернативные традиционным ДР ПЛИС (например, в [57] рассмотрен оптоэлек-

тронный ДРМ). Таким образом, необходимо искать общие математические модели, которые инварианты по отношению к элементной базе и отражают фундаментальную для вычислительной техники сущность процесса динамической реконфигурации.

Следует подчеркнуть, что в области ДРС мало общепризнанных математических моделей. Более того, в области теоретических обобщений они практически отсутствуют. Данный факт обусловлен как сложностью решаемой задачи, так и новаторством рассматриваемого направления. В качестве первого шага на пути к созданию полноценного аппарата моделирования ДРС были предложены два типа математических моделей ДРС [49, 51]:

− ***модель «статических вычислений»*** предназначена для математического описания процессов, происходящих в ***стационарных ДРС***, то есть в таких ДРС, у которых набор команд, а также структура вычислительных модулей, реализующих команды ДРС, и условия их выполнения (наличие или отсутствие конвейера при выполнении команд; количество одновременно выполняемых команд и потоков данных; тип параллелизма) остаются постоянными в течение работы вычислителя;

− ***модель «динамических вычислений»*** представляет собой более общий случай по сравнению с моделью «статических вычислений» и служит для описания ***нестационарных ДРС***, то есть таких ДРС, у которых набор команд $\mathrm{A}^{\Delta t_k}$ (точнее, его подмножество $\mathbf{A}_{\mathrm{adapt}}^{\Delta t_k}$) или структура вычислительных модулей, реализующих команды ДРС, или условия их выполнения изменяются с течением времени.

Стационарные ДРС фактически представляют собой вычислители, построенные на принципах фон Неймана. Нестационарные ДРС, напротив, являются наиболее общим случаем для вычислительных устройств, обладающих свойством динамической реконфигурации. Подробное обоснование такой классификации ДРС, а также пояснения к терминам, используемым в определениях, приведены в [49, 51]. Модель «статических вычислений» выделена особо в виду прагматических соображений: класс стационарных ДРС довольно многочисленный, а математическая модель для него как для частного случая

нестационарных ДРС априорно представляется существенно более простой, чем для общего случая. Это позволит значительно сократить временные затраты как на разработку и реализацию самой математической модели, так и на собственно процесс моделирования, снизив его вычислительную сложность [49, 51].

Первоочередным вопросом, возникающим в контексте задачи математического моделирования ДРС, является то, оценки каких величин требуется получить в процессе моделирования. На него можно ответить, обратившись к практическим потребностям разработчиков электронных средств. Так, для разработки эффективного компилятора абсолютно необходимы данные о времени выполнения команд [2]. Исходя из этого, получение оценок времени выполнения отдельных задач и групп задач на базе ДРС можно считать основной целью математического моделирования таких систем. Следующий шаг – это анализ того, какие структурные элементы модели существуют уже в настоящее время, а какие отсутствуют. Для контроллера ДРС, то есть традиционного фон-неймановского процессора или его подмножества, общей математической модели не существует, но она может быть получена для каждого конкретного экземпляра отдельно. Для ДРМ, то есть ДР ПЛИС, известны модель процесса конфигурирования ДР ПЛИС с полной реконфигурацией [30] и ее обобщенный вариант на случай применения ДР ПЛИС с частичной реконфигурацией [48]. Таким образом, построение математической модели, пригодной для разработки компилятора с языка высокого уровня под ДРС, ограничивается лишь отсутствием модели процесса взаимодействия элементов ДРС между собой [49, 51].

В частности, математические модели традиционных процессоров и стационарных ДРС отличаются только тем, что в первом случае время выполнения операции не зависит от числа ее повторений (если исключить зависимость, связанную с функционированием конвейера). Во втором случае эта зависимость существует, и ее следует принимать во внимание при разработке компилятора, поскольку одну и ту же операцию можно реализовать как на базе контроллера, так и на базе ДРМ. Однако в том случае, когда большое количество однотипных

операций необходимо обрабатывать параллельно, могут задействоваться оба модуля ДРС одновременно. Следовательно, время конфигурирования ДРМ можно не принимать во внимание, если выполняется группа условий:

$$\begin{cases} n_{\|}^{\text{real}} \geq n_{\|}^{\text{Pr}} \\ N_{\text{reit}} \geq N_{\text{thr}} \end{cases}, \tag{3.1}$$

где $n_{\|}^{\text{real}}$ – число одинаковых операций, которое необходимо выполнять параллельно; $n_{\|}^{\text{Pr}}$ – количество одинаковых операций, которые могут выполняться в контроллере ДРС одновременно; N_{reit} – число повторений операции; $N_{\text{thr}} = f\left(T_{\text{CONF}}, n_{\|}^{\text{Pr}}\right)$ – пороговое значение, превышение которого означает, что для выполнения операций целесообразно задействовать контроллер и ДРМ одновременно. Следует подчеркнуть, что величина N_{thr} может быть вычислена заранее (до процесса компиляции) и храниться как параметр [49, 51].

Возвращаясь к выбору между реализацией на базе ДРМ и контроллера ДРС, необходимо определить среднее время выполнения некоторой произвольной операции. Оно рассчитывается как

$$t_{\text{op}}^{\text{ACS}} = f\left(N_{\text{reit}}\right) = \frac{T_{\text{CONF}}}{N_{\text{reit}}} + f_{\text{Arch}}^{\text{DRM}}\left(t_{\text{op}}^{\text{real}}, N_{\text{reit}}\right), \tag{3.2}$$

где T_{CONF} – время, необходимое для конфигурирования ДРМ перед началом выполнения операции; $t_{\text{op}}^{\text{real}}$ – время выполнения операции на ДРМ; $f_{\text{Arch}}^{\text{DRM}}(\cdot)$ – функция, определяющая архитектурные особенности реализации вычислений на базе ДРМ (например, особенности конвейерной архитектуры). На базе процессора та же самая операция (в случае, если она не эквивалентна одной из команд) выполняется за время, равное

$$t_{\text{op}}^{\text{Pr}} = f_{\text{Arch}}^{\text{Pr}}\left(\left\{t_{\text{op}i}\right\}_{i=0}^{r-1}\right), \tag{3.3}$$

где $f_{\text{Arch}}^{\text{Pr}}(\cdot)$ – функция, определяющая архитектурные особенности реализации вычислений на базе процессора (например, особенности его конвейерной архитектуры; $t_{\text{op}i}$ – время выполнения i-й команды процессора, необходимой для выполнения заданной операции; r – общее

число команд процессора, необходимых для выполнения заданной операции;. Условием выполнения операции на базе ДРМ является

$$t_{\text{op}}^{\text{Pr}} > t_{\text{op}}^{\text{ACS}}. \tag{3.4}$$

Подставляя (3.2) и (3.3) в (3.4), получаем

$$f_{\text{Arch}}^{\text{Pr}}\left(\left\{t_{\text{op}i}\right\}_{i=0}^{r-1}\right) > \frac{T_{\text{CONF}}}{N_{\text{reit}}} + f_{\text{Arch}}^{\text{DRM}}\left(t_{\text{op}}^{\text{real}}, N_{\text{reit}}\right). \tag{3.5}$$

При достаточно малом числе операций, выполняемых подряд, N_{reit} может оказаться нецелесообразным использование ДРМ, и операции будут реализовываться программно на базе контроллера ДРС. Уровень формализации задачи, представленный выше, вполне достаточен, чтобы разработать компилятор для стационарных ДРС. В случае нестационарных ДРС необходимо анализировать сценарий вычислительного процесса, то есть рассматривать временную диаграмму работы ДРС (как сделано, например, в [51]) [49].

Следует заметить, что в перспективе компилятор должен сам выбирать модель ДРС, которая соответствовала бы имеющемуся аппаратному обеспечению и вектору решаемых задач, то есть коду, введенному на языке программирования высокого уровня. Рассмотренные выше модели и методики позволяют сделать первый шаг на пути к достижению этой цели.

3.2. Особенности выполнения расчетов на базе динамически реконфигурируемых систем обработки информации

Процесс программирования и выполнения расчетов на базе ДРС имеет свою специфику, которая обусловлена архитектурными особенностями подобных вычислительных устройств, а именно адаптивностью системы команд ДРС. Как уже было сказано выше, под этим термином («адаптивность системы команд») понимается не только добавление функционально новых команд (например, в составе системы команд управляющего модуля имелось две команды – вычисление функций $\sin(x)$ и $\cos(x)$, в дополнение к ним на базе ДРМ может быть реализована команда для одновременного расчета обеих функций) или замена одних команд на другие, но и изменение принципов

выполнения команд (в частности, реализация на базе ДРМ SIMD-команды, функционально аналогичной SISD-команде, имеющейся в составе управляющего модуля ДРС). В этой связи одной из ключевых задач становится сравнение времени выполнения одних и тех же операций на базе управляющего модуля и других процессоров, входящих в состав ДРС (например, цифрового сигнального процессора), с одной стороны, и на базе ДРМ, с другой. Следует подчеркнуть, что подобные сравнения не всегда могут быть выполнены в процессе компиляции, то есть заранее. В ряде случаев (как правило, при обработке массивов данных переменной длины) возникает необходимость в таких процедурах непосредственно в процессе выполнения кода, что ведет к дополнительным вычислительным затратам, которые тоже должны быть учтены в математической модели ДРС [47].

В этой связи ключевым моментом проектирования ДРС является выбор системы команд, точнее, адаптируемой ее части $\mathbf{A}_{\text{adapt}}^{\Delta t_k}$. Адекватный выбор подсистемы команд $\mathbf{A}_{\text{adapt}}^{\Delta t_k}$ позволяет существенно сократить временные затраты на вычисления – как на ***основные*** (обусловленные самим вычислительным процессом), так и на ***вспомогательные***, то есть на те, которые обусловлены дуальной (для простых ДРС) или гибридной (для гибридных ДРС) архитектурой вычислителя. В этой связи для разработчиков вычислительных устройств целесообразно сформулировать основные правила выбора подсистемы команд $\mathbf{A}_{\text{adapt}}^{\Delta t_k}$, которые следуют из принципов построения ДРС. К таким правилам относятся следующие [47]:

– ДРМ следует проектировать только как SIMD- или как MIMD-вычислитель, поскольку это позволяет наиболее полно использовать возможности параллельной обработки данных;

– при выборе между функционально одинаковыми композицией элементарных операций и составной операцией (значение термина «составная операция» подробно раскрыто в [7]) целесообразно отдавать предпочтение последней, потому что указанный подход позволяет сократить как число обращений к памяти, так и длительность самих расчетов;

– в случае необходимости передачи больших объемов данных, для выборки которых из памяти применяется специфический алгоритм (в частности, при выборке данных для алгоритмов сжатия изображений SPIHT или видеопоследовательностей 3D-SPIHT, описанных соответственно в [70] и [66]), целесообразно рассмотреть возможность разработки специализированного DMA-контроллера (Direct Memory Access, прямой доступ к памяти), что позволяет в полной мере использовать преимущества перепрограммируемой элементной базы.

Очевидно, что сложность математической модели ДРС зависит от того, насколько качественно была проведена декомпозиция процесса обработки данных. В основе предлагаемой модели лежит принцип разбиения всего вычислительного процесса, протекающего на базе ДРС, на так называемые макрооперации, позволяя применять данное понятие к широкому кругу ДРС, что дает основания говорить об адекватности такого варианта декомпозиции и об универсальности рассматриваемой ниже математической модели [47].

Макрооперация – это логически взаимосвязанная совокупность элементарных операций, выполняемых с использованием управляющего и динамически реконфигурируемого модулей ДРС, причем за время выполнения указанной совокупности операции ДРМ конфигурируется не более одного раза. Макрооперация является естественной величиной для оценки сложности алгоритма, решаемого на базе ДРС. Следует подчеркнуть, что макрооперация – это интегральная величина, которая позволяет оценить как основные, так и вспомогательные временные затраты. Именно поэтому целесообразно её использование при решении задачи математического моделирования ДРС [47].

Другим важным моментом является то, каким образом производится выполнение макроопераций на базе ДРМ [47]:

1) управляющий модуль производит выборку и декодирование макрокоманды, которая указывает адрес конфигурационного файла в памяти и инициирует его загрузку в ДРМ; под макрокомандой понимается совокупность команд управляющего модуля; в предельном случае операция загрузки конфигурационного файла в ДРМ может быть реализована в виде отдельной специальной команды; длительность выполнения данного этапа обозначим через T_{DEC};

2) происходит конфигурирование ДРМ, длительность которого равна T_{CONF}; в случае, если ДРМ поддерживает режим частичного конфигурирования (то есть смена конфигурации производится не для всего ДРМ, а только для одной или нескольких его частей), то значение T_{CONF} меняется в зависимости от информации, содержащейся в конфигурационном файле: $T_{\text{CONF}}(n_{CF}) \neq const$, где n_{CF} – номер загружаемого конфигурационного файла;

3) после конфигурирования ДРМ начинает выполнение расчетов, которое заключается в выборке из памяти исходных данных и выполнении вычислений. Время выборки данных t_{FETCH}, как и время вычисления t_{CALC} в общем случае не являются постоянными, а изменяются от итерации к итерации, то есть $t_{\text{FETCH}}(i) \neq const$ и $t_{\text{CALC}}(i) \neq const$, где i – номер итерации при выполнении одной и той же операции на базе ДРМ без реконфигурирования.

Среднее время $\overline{t_{\text{ITER}}}$ выполнения одной итерации, которое и является одним из критериев, применяемых для оценки эффективности ДРС при функционировании ДРМ в режиме SIMD-вычислителя, есть функция трех величин: $\overline{t_{\text{ITER}}} = f(t_{\text{FETCH}}, t_{\text{CALC}}, N_{\text{REIT}})$, где N_{REIT} – число выполнений одной и той же операции между двумя ближайшими процедурами конфигурирования ДРМ. Вид функции $\overline{t_{\text{ITER}}} = f(t_{\text{FETCH}}, t_{\text{CALC}}, N_{\text{REIT}})$ определяется логикой работы реализованного конечного автомата. В частности, если выборка и вычисления следуют друг за другом строго последовательно, то согласно [47]

$$\overline{t_{\text{ITER}}} = f(t_{\text{FETCH}}, t_{\text{CALC}}, N_{\text{REIT}}) = \frac{\sum_{i=0}^{N_{REIT}-1} (t_{\text{FETCH}}(i) + t_{\text{CALC}}(i))}{N_{\text{REIT}}}. \qquad (3.6)$$

Если $t_{\text{FETCH}}(i) = t_{\text{FETCH}} = const$ и $t_{\text{CALC}}(i) = t_{\text{CALC}} = const$, то можно упростить выражение (3.6) и перейти к функции двух переменных [47]:

$$\overline{t_{\text{ITER}}} = f(t_{\text{FETCH}}, t_{\text{CALC}}, N_{\text{REIT}}) = t_{\text{FETCH}} + t_{\text{CALC}}. \qquad (3.7)$$

Если конечный автомат построен по конвейерному принципу, то есть во время выполнения вычислений производится выборка данных для следующей итерации, то согласно [47]

$$\overline{t_{\text{ITER}}} = \frac{t_{\text{FETCH}}(0) + \sum_{i=0}^{N_{\text{REIT}}-2} \max\{t_{\text{CALC}}(i), t_{\text{FETCH}}(i+1)\} + t_{\text{CALC}}(N_{\text{REIT}}-1)}{N_{\text{REIT}}}. \quad (3.8)$$

Таким образом, общее время выполнения макрооперации на базе ДРМ, функционирующего в режиме SIMD-вычислителя, рассчитывается как

$$T_{\text{DRM\_SIMD}} = T_{\text{CONF}}(n_{\text{CF}}) + N_{\text{REIT}} \cdot \overline{t_{\text{ITER}}}. \quad (3.9)$$

Оценка еще больше усложняется, если ДРМ работает как MIMD-вычислитель, который может быть реализован двумя различными способами [47]:

1) как совокупность логически независимых SIMD-вычислителей, которые не обмениваются данными друг с другом;

2) как совокупность синхронизованных SIMD-вычислителей (рис. 2.21), которые обмениваются данными (как промежуточными, так и конечными) друг с другом в процессе работы.

Очевидно, что в первом случае можно использовать те же самые модели, которые были предложены для SIMD-вычислителей. Во втором случае это, к сожалению, не так. Принимая во внимание тот факт, что команды в MIMD-вычислителе выполняются параллельно, можно записать следующее выражение:

$$T_{\text{DRM\_MIMD}} = T_{\text{CONF}}(n_{\text{CF}}) + \max_{j=0,N_{\text{INST}}-1}\left\{N_{\text{REIT}}(j) \cdot \overline{t_{\text{ITER}}(j)}\right\}, \quad (3.10)$$

где N_{INST} – число команд, выполняемых параллельно ДРМ в режиме MIMD-вычислителя. Время, затрачиваемое одним вычислительным процессом на ожидание промежуточных или окончательных результатов выполнения другого вычислительного процесса, включается в значение величины $\overline{t_{\text{ITER}}}$. Доля времени ожидания в общем значении величины $\overline{t_{\text{ITER}}}$ определяется конкретной реализацией конфигурационного файла для ДРМ [47].

Для простейшего случая, когда ДРС состоит из одного управляющего модуля и одного ДРМ (см. рис. 1.1), общее время выполнения макрооперации рассчитывается как

$$T_{\text{MACRO}} = \max\{T_{\text{CM}} + T_{\text{DEC}}, T_{\text{DRM}}\}, \quad (3.11)$$

где $T_{\text{СМ}}$ – время, затрачиваемое управляющим модулем на выполнение макрооперации. Оно определяется следующими факторами [47]:

– ***структурой макрооперации***, то есть тем, из каких элементарных операций она состоит, какие типы параллелизма заложены в неё и т. д.;

– ***архитектурой управляющего модуля***: фактически это архитектура процессора, на базе которого построен управляющий модуль (например, векторная, матричная, систолическая, суперскалярная архитектуры).

Для случая, когда ДРС состоит из одного управляющего модуля, N_{PROC} дополнительных процессоров, которые могут иметь различную архитектуру, и одного ДРМ, общее время может быть вычислено по формуле:

$$T_{\text{MACRO}} = \max\{T_{\text{СМ}} + T_{\text{DEC}}, \max_{j=0,N_{\text{PROC}}-1}\{T_{\text{PROC}}(j)\}, T_{\text{DRM}}\}, \qquad (3.12)$$

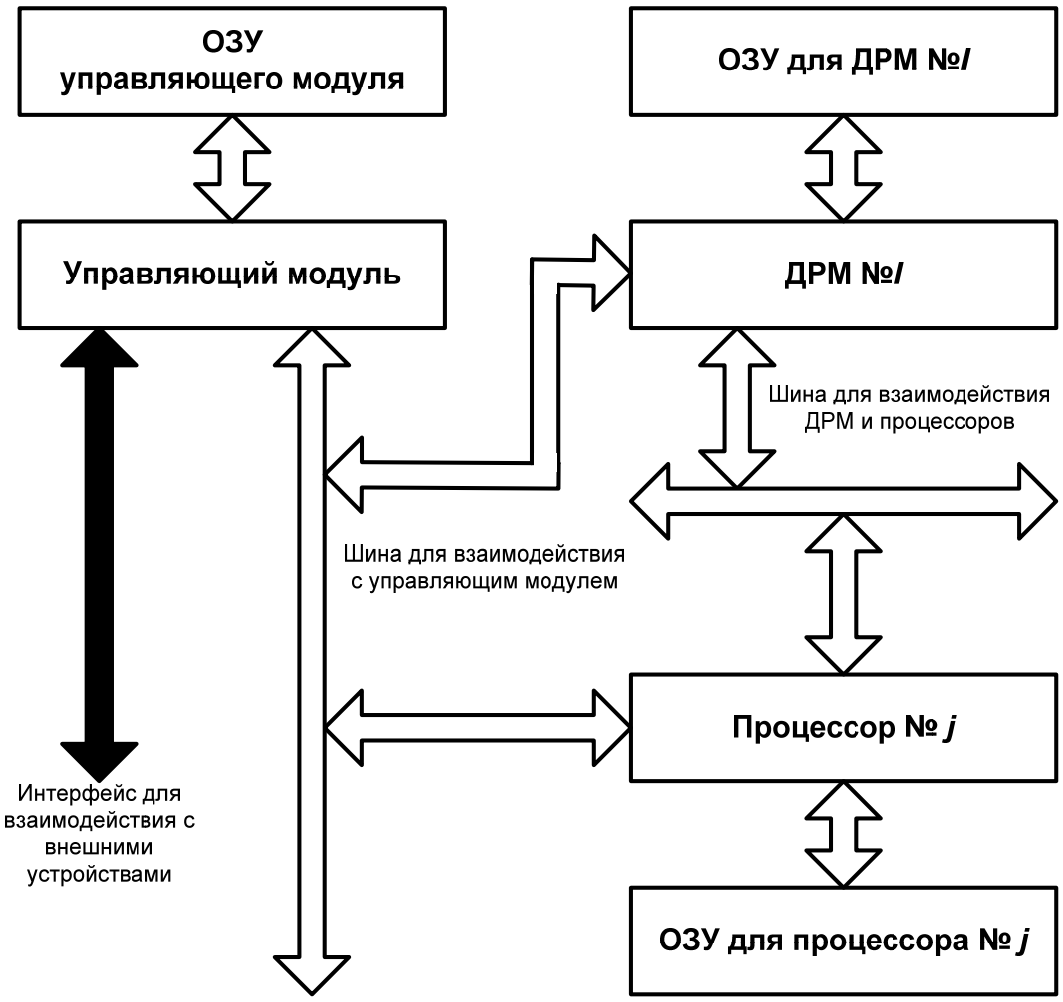
где $T_{\text{PROC}}(j)$ – время выполнения части макрооперации на *j*-м процессоре [47].

Следует подчеркнуть, что понятие макрооперации напрямую применимо только к ДРС, включающим один ДРМ, поскольку в этом случае все макрооперации будут исполняться строго последовательно. Для ДРС, состоящих из двух и более ДРМ, вводится понятие «виртуальной ДРС», которое позволяет, используя принцип декомпозиции, применить понятие макрооперции и к ним. ***Виртуальная ДРС*** – это ДРС, условно выделяемая в составе реальной ДРС, включающей более двух ДРМ, и содержащая не более одного управляющего модуля и одного ДРМ. Таким образом, любая ДРС может быть рассмотрена как совокупность элементарных ДРС и традиционных процессоров. Продемонстрировать принцип использования декомпозиции ДРС, содержащих два и более ДРМ, можно на примере ДРС, показанной на рисунке. Данная ДРС состоит из одного управляющего модуля, N_{DRM} ДРМ и N_{PROC} процессоров. Причем каждая ДРМ и каждый процессор подключаются одновременно к двум шинам [47]:

– шине взаимодействия с управляющим модулем, которая предназначена, прежде всего, для получения ДРМ или процессорами

доступа к данным, содержащимся в ОЗУ управляющего модуля, или для получения управляющим модулем доступа к данным, содержащимся в ОЗУ ДРМ или процессоров;

– шине взаимодействия ДРМ и процессоров, которая предназначена для снижения нагрузки на шину взаимодействия с управляющим модулем и на сам управляющий модуль при получении ДРМ или процессорами доступа к данным, содержащимся в ОЗУ других ДРМ или процессоров.



Обобщенная структурная схема ДРС, состоящей из одного управляющего модуля, N_{DRM} ДРМ и N_{PROC} процессоров

В этом случае время выполнения макрооперации на l-й виртуальной ДРС вычисляется по формуле, аналогичной (3.12):

$$T_{\mathrm{MACRO}}(l) = \max\{T_{\mathrm{CM}}(l) + T_{\mathrm{DEC}}(l), \max_{j=0,N_{\mathrm{PROC}}-1}\{T_{\mathrm{PROC}}(j,l)\}, T_{\mathrm{DRM}}(l)\}, \qquad (3.13)$$

где $T_{\text{CM}}(l)$, $T_{\text{DEC}}(l)$ и $T_{\text{DRM}}(l)$ – время, затрачиваемое, соответственно, управляющим модулем на декодирование и выборку команды, а также на выполнение макрооперации и динамически реконфигурируемым модулем на выполнение макрооперации. Выражения (3.6) – (3.13) могут быть уточнены, если используются специальные режимы обработки и передачи данных (в частности, прямой доступ к памяти) [47].

Следует подчеркнуть, что значения величин $T_{\text{CM}}(l)$, $T_{\text{DRM}}(l)$ и $T_{\text{PROC}}(j,l)$ взаимосвязаны, поскольку при решении одной задачи на параллельном вычислителе возникает проблема обмена данными между параллельно выполняемыми подзадачами. При этом для минимизации общего времени решения задачи необходимо обеспечить:

− *на этапе декомпозиции задачи и распределения полученных подзадач по различным вычислительным модулям (partitioning task)* максимально одинаковые значения времени выполнения каждой из подзадач;

− *на этапе разработки конфигурационных файлов для ДРМ, а также написания программного обеспечения для процессоров и управляющего модуля* синхронизацию между подзадачами с целью минимизации времени ожидания одним вычислительным модулем данных от другого вычислительного модуля [47].

Кроме того, при математическом моделировании ДРС, содержащих два и более ДРМ, возникают две большие сложности [47]:

− макрооперации внутри одной ДРС (на различных виртуальных ДРС) могут выполняться параллельно;

− процессоры, входящие в состав ДРС, могут, в общем случае, участвовать одновременно в выполнении нескольких макроопераций, то есть могут одновременно входить в состав нескольких виртуальных ДРС.

Если параллельное выполнение нескольких макроопераций является безусловным преимуществом ДРС, то этого нельзя столь безоговорочно утверждать касательно вовлечения процессоров в одновременное выполнение нескольких макроопераций. Это связано, прежде всего, с тем, что при переходе от одной макрооперации к другой происходит смена ветви выполнения команд (фактически, безусловный переход), резко снижая эффективность работы предсказателя и обу-

словливая наличие холостых циклов у процессора (так называемых NOP – «No OPerations»), а следовательно, и сбои в работе конвейера. Влияние подобных негативных эффектов может быть минимизировано, если снизить частоту переходов от одной макрооперации к другой и увеличить число команд, выполняемых подряд в пределах одной макрооперации. Частным случаем такого подхода является постоянное включение того или иного процессора в состав одной и той же виртуальной ДРС, то есть процессор будет принимать участие в выполнении только одной макрооперации [47].

Время выполнения задачи, разбитой на несколько макроопераций, параллельно выполняемых на базе различных виртуальных ДРС, рассчитывается как

$$T_{\text{TSK}} = \max_{l=\overline{0,N_{DRM}-1}} \{T_{\text{MACRO}}(l) + T_{\text{OFFSET}}(l)\}, \qquad (3.14)$$

где N_{DRM} – число ДРМ в составе ДРС, $T_{\text{OFFSET}}(l)$ – время смещения начала выполнения макрооперации на l-й виртуальной ДРС относительно начала выполнения макрооперации на 0-й виртуальной ДРС. В качестве 0-й виртуальной ДРС, как правило, выбирается та, на которой раньше других начинается выполнение задачи. Таким образом, $T_{\text{OFFSET}}(0) = 0$ и $T_{\text{OFFSET}}(l) \geq 0$ для $l = \overline{1, N_{DRM} - 1}$ [47].

Наиболее практически значимой математической моделью будет та, которая позволит описать режим работы ДРС, включающий произвольное число макроопераций, выполняемых как последовательно, так и параллельно. В этом случае время решения задачи будет описываться следующим уравнением:

$$T_{\text{TSK}} = \max_{l=\overline{0,N_{DRM}-1}} \left\{ \sum_{k=0}^{N_{MACRO}(l)-1} \left(T_{\text{MACRO}}(k,l) + T_{\text{DELAY}}(k,l) \right) \right\}, \qquad (3.15)$$

где $N_{\text{MACRO}}(l)$ – число макроопераций, выполняемых на l-й виртуальной ДРС при решении рассматриваемой задачи; $T_{\text{MACRO}}(k,l)$ – время выполнения k-й макрооперации на l-й виртуальной ДРС; $T_{\text{DELAY}}(k,l)$ – время задержки начала выполнения k-й макрооперации относительно $(k-1)$-й макрооперации на l-й виртуальной ДРС. Очевидно, что при $k = 0$ $T_{\text{DELAY}}(0,l) = T_{\text{OFFSET}}(l)$ и выбирается таким же образом, как это было указано для (3.14) [47].

Таким образом, рассмотренные математические модели ДРС позволяют рассчитывать время выполнения задач на различных архитектурах. Безусловно, данные модели не являются абсолютно универсальными, поскольку в ряде случаев (прежде всего, при использовании специальных режимов обработки данных − таких, как DMA) требуют дополнительного уточнения. Тем не менее подобные математические модели крайне важны для понимания процессов, происходящих внутри ДРС, и могут служить основой для создания математического аппарата, описывающего специфические архитектуры ДРС [47].

Вопросы и задания для самоподготовки

1. Что является основной целью математического моделирования ДРС? Для чего могут применяться модели ДРС? В чем состоит их прикладное значение?

2. Математические модели каких вычислительных устройств используются в качестве исходных при разработке математической модели ДРС в целом? Объясните, почему. С какими моделями Вы были знакомы до изучения данного курса?

3. Какие типы математических моделей ДРС Вам известны? Почему для моделирования ДРС целесообразно использовать различные типы моделей? Чем они отличаются друг от друга?

4. Что такое стационарные ДРС? Чем они отличаются от нестационарных ДРС? Приведите примеры ДРС обоих типов. Обоснуйте Ваше мнение.

5. Какое влияние оказывает такой феномен, как динамическая реконфигурация, на процесс моделирования ДРС? Каким образом учитывается время конфигурирования ДРМ в математических моделях ДРС? Всегда ли необходимо учитывать этот параметр в математических моделях ДРС?

6. Перечислите основные правила выбора команд, включаемых в состав адаптируемой подсистемы команд ДРС. Обоснуйте эти правила математически.

7. Дайте определение понятию «макрооперация». Какова роль макроопераций в создании математических моделей ДРС? Опишите алгоритм выполнения макрооперации на базе ДРМ.

8. Приведите аналитические выражения, предназначенные для оценки различных временных характеристик, описывающих длительность выполнения макроопераций на базе ДРМ с различными типами организации конечного автомата. В каких случаях и каким именно образом данные формулы могут быть использованы на практике?

9. Сравните оценки временных характеристик для SIMD- и MIMD-архитектур вычислителя, построенного на базе ДРМ. Как использование MIMD-архитектуры сказывается на производительности ДРС?

10. Что такое «виртуальная ДРС»? Как можно использовать данную абстракцию для математического моделирования сложных ДРС? Приведите примеры применения подобных моделей, а также формализованные оценки временных характеристик для сложных ДРС.

ГЛАВА 4. ПРИМЕРЫ ПРИМЕНЕНИЯ ДИНАМИЧЕСКИ РЕКОНФИГУРИРУЕМЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ

В данной главе рассматриваются практические примеры применения динамически реконфигурируемых систем обработки информации. Основное внимание уделено таким приложениям, которые в наибольшей степени подчеркивают преимущества ДРС по сравнению с традиционными решениями. В частности, приведены примеры использования ДРС для построения устройств адаптивного вычисления математических функций и адаптивного сжатия данных. Реализация подобных систем на базе традиционных вычислителей (процессоров или заказных микросхем) или крайне затруднительна, или требует существенно больших материальных затрат, чем для ДРС.

4.1. Устройство адаптивного вычисления математических функций

Расчет математических функций – как элементарных (например, тригонометрических, логарифмических, экспоненциальных, показательных), так и составных – является одной из наиболее распространенных операций в вычислительной технике. Именно поэтому указанная задача показательна при сравнении результатов, полученных на базе традиционных вычислителей и ДРС. В настоящей главе рассмотрен пример вычисления прямых тригонометрических функций (ПТФ), к которым относятся синус $\sin x$, косинус $\cos x$, тангенс $\mathrm{tg}\,x$ и котангенс $\mathrm{ctg}\,x$, поскольку они в настоящее время находят самое широкое применение в системах управления и обработки информации у различных изделий, выпускаемых промышленностью, – в станках с ЧПУ (при решении траекторных задач, сложном формообразовании и трехмерном моделировании) [3 − 7, 15], в системах цифровой обработки сигналов [27, 35] и компрессии

мультимедийных данных [14], телекоммуникационных, радиолокационных и радионавигационных устройствах, аэрокосмических аппаратах (при коррекции орбиты) [22] и т.д. При выполнении указанных задач в режиме реального времени часто возникает необходимость в повышении скорости обработки информации, что требует сокращения времени вычисления ПТФ (так как даже в цифровых сигнальных процессорах оно превышает время умножения приблизительно в 25 раз [27]) и достигается аппаратной реализацией алгоритмов расчета данного типа функций. Однако классические варианты (на основе стандартных, заказных или полузаказных микросхем) не лишены недостатков, главным из которых является их ориентация на довольно узкий спектр вычислительных методов и разрядностей данных. В предельном случае подобные операции реализуются на основе одного из численных алгоритмов и рассчитаны на использование единственного формата данных. Такой подход к вычислениям оправдан только при выполнении двух условий [48]:

– требования к точности расчетов остаются постоянными при решении любых задач, для которых предназначено данное устройство обработки информации;

– отсутствует необходимость в изменении используемого математического аппарата.

На практике эти условия не всегда выполняются, что во многих случаях приводит к дополнительным временным затратам. Следует учесть, что при изменении точности недостаточно просто изменить разрядность данных, – для достижения лучшего результата может потребоваться замена алгоритма расчета (в частности, при разрядностях 8-12 бит целесообразно использование табличного метода, при разрядностях 32-64 бита – итерационных методов). Пример того, каким образом и в каком порядке следует менять итерационные методы при вычислении ПТФ, был рассмотрен в [46]. Краткие сведения, характеризующие изменения времени расчета ПТФ в зависимости от разрядности данных, приведены в таблице.

Таблица

Зависимость времени расчета ПТФ от разрядности данных для различных методов

Метод	Аналитически заданная зависимость	Характер зависимости
Группа методов «цифра за цифрой»	$t_{calc} \approx 10nt_{add}$	Линейная
Метод численного решения дифференциальных уравнений (на основе метода Эйлера)	$t_{calc} \approx 2^{n/2} t_{add}$	Степенная
Метод сужения интервала	$t_{calc} \approx 4n^2 t_{add}$	Квадратичная

Примечание: t_{add} – время выполнения операции сложения.

Рис. 4.1 наглядно иллюстрирует зависимости, записанные аналитически в таблице.

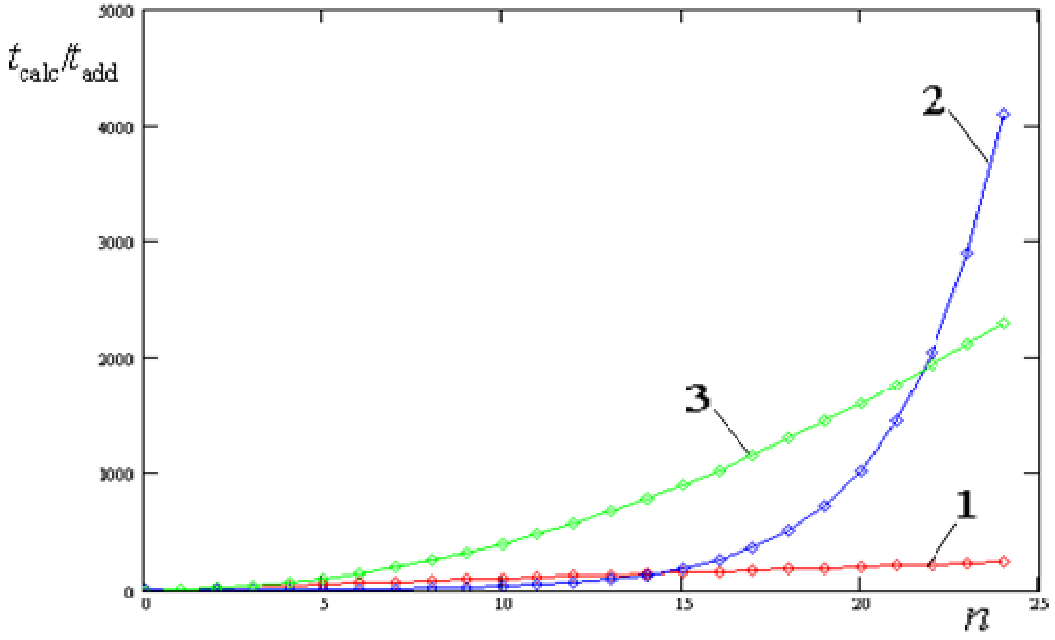


Рис. 4.1. Графики зависимостей величины t_{calc} / t_{add} от разрядности данных n для различных методов: 1 – для группы методов «цифра за цифрой», 2 – для метода численного решения дифференциальных уравнений (на основе метода Эйлера), 3 – для метода сужения интервала

Кроме того, может меняться набор задач, решаемых вычислителем, а следовательно, и набор выполняемых операций, к которым предъяв-

ляется не только требование функциональной полноты, но и минимизации временных затрат. В классических системах обработки информации это может привести к замене аппаратного обеспечения, т.е. к значительным материальным затратам. Таким образом, возникает потребность в разработке альтернативных устройств расчета ПТФ [48].

Подобной альтернативой являются системы обработки информации, которые отличаются от классических аналогов тем, что способны менять свою структуру в процессе функционирования в зависимости от условий решаемой задачи (точности, наличия или отсутствия априорной информации об изменении аргумента и т.д.), то есть могут адаптироваться к ним. Указанный подход позволяет выбрать такие способы представления данных и алгоритмы вычисления ПТФ, которые в наибольшей степени соответствуют требованиям, предъявляемым в данный момент. Следовательно, наиболее подходящей элементной базой для реализации адаптивного устройства вычисления ПТФ являются ДР ПЛИС, допускающие неограниченно большое число замен конфигураций непосредственно в процессе работы [48].

Однако в соответствии с принципом системного подхода для сокращения времени расчета ПТФ необходимо совершенствовать не только техническое, но и математическое обеспечение. По этой причине в научно-технической литературе уделяется большое внимание численным алгоритмам и вычислительным структурам, реализующим указанные алгоритмы. Следует отметить, что между системами обработки информации и используемым в них математическим аппаратом существует неразрывная связь. Для систем адаптивного вычисления ПТФ наличие указанной взаимосвязи еще более очевидно, так как непосредственно пользователь (а не только разработчик) может изменить набор вычислительных операций и тем самым повлиять на быстродействие системы [48].

Таким образом, исследования по созданию новых систем обработки информации, требующих расчета ПТФ, должны вестись с применением методов системного анализа, причем одновременно в двух направлениях:

– совершенствование математического обеспечения (численных методов и алгоритмов),

– поиск таких технических реализаций, которые позволяют минимизировать время расчета ПТФ для каждой из решаемых задач.

Оба направления одинаково важны и актуальны, ни одно из них не может быть признано более приоритетным [48].

Обработка информации в системе адаптивного вычисления ПТФ, как и в любых других вычислительных системах, представляет собой процесс с определенными, четко выраженными иерархическими уровнями. Каждый уровень характеризуется тем, какие единицы информации на нем обрабатываются, какие операции выполняются над данными единицами информации и в каких устройствах реализуются данные операции. При этом операции верхнего уровня являются композициями операций нижнего уровня, а устройства, в которых они выполняются, – системами, состоящими из устройств нижнего уровня. Таким образом, сложность единиц информации, операций и вычислительных устройств повышается при переходе от младшего уровня к старшему [29]. Следовательно, система адаптивного вычисления ПТФ является объектом системного анализа, потому что обладает всеми основными признаками сложной системы, указанными в [54]:

– включает в себя большое число разнородных элементов (устройств расчета ПТФ, которые построены на основе различного математического обеспечения и могут использовать различные типы управления);

– сложность функций, выполняемых системой (система производит вычисление ПТФ при изменяющихся условиях решения задач, что требует анализа условий задач, выбора алгоритма расчета ПТФ и максимально быстрого вычисления функций);

– сложный характер, неоднородность связей между подсистемами (следует из функций, выполняемых системой: связи между операционными и управляющими автоматами устройств расчета ПТФ, входящих в ее состав, могут отличаться, причем довольно значительно);

– наличие неопределенности в описании системы (неопределенность возникает, например, при описании временных характеристик системы и составлении алгоритма адаптации);

– сложность определения требуемого управляющего воздействия на систему (различные устройства вычисления ПТФ, входящие в состав адаптивной системы, могут требовать применения различных

типов управления, что приводит к необходимости построения достаточно сложного управляющего автомата) [48].

Для построения математической модели системы необходимо произвести ее декомпозицию. Декомпозицию сложных дискретных систем, к которым относится и система адаптивного вычисления ПТФ, можно производить на основе теории автоматов. В этом случае устройство вычисления ПТФ представляется в виде двух взаимодействующих автоматов – операционного и управляющего. Операционный автомат реализует шаги алгоритма, а управляющий автомат определяет порядок выполнения шагов алгоритма [17, 18]. Применительно к системе адаптивного вычисления ПТФ *операционный автомат* реализует алгоритмы, предназначенные непосредственно для расчета функций (*вычислительные алгоритмы*). Управляющий автомат выполняет общие функции управления системой, а также производит выбор вычислительных алгоритмов в соответствии с точностью выполняемых расчетов и известной априорной информацией о законе изменения аргументов ПТФ, что определяется *алгоритмом адаптации* (алгоритмом адаптации к точности вычислений). Результатом декомпозиции являются обобщенная структурная схема и граф состояний системы адаптивного вычисления ПТФ, представленные на рис. 4.2 и 4.3 [48].

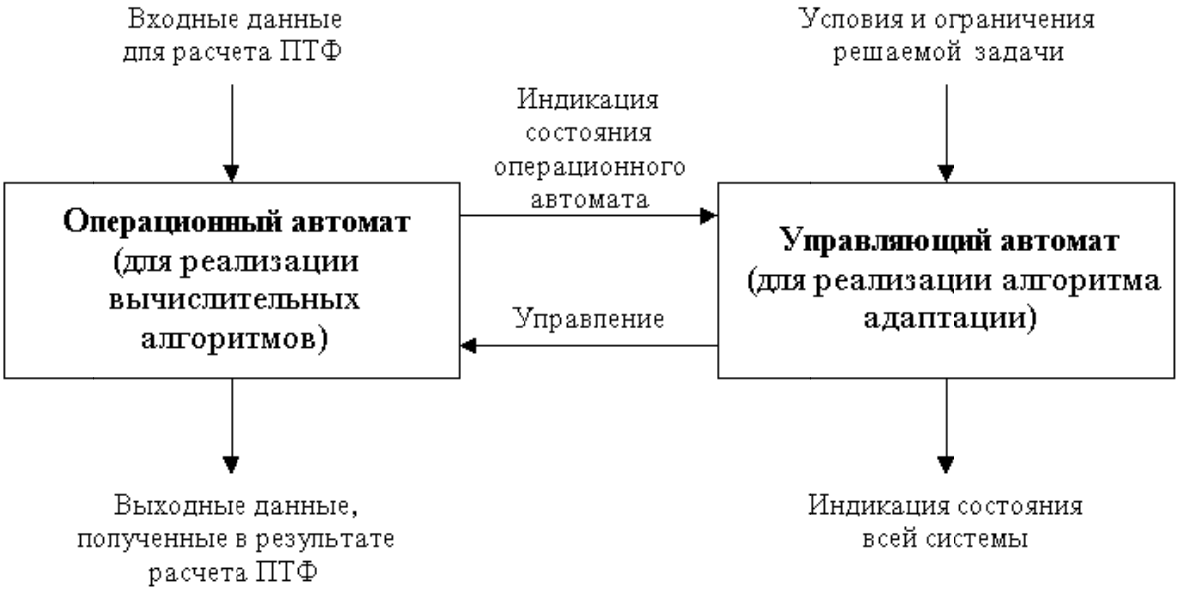


Рис. 4.2. Обобщенная структурная схема адаптивного устройства вычисления ПТФ

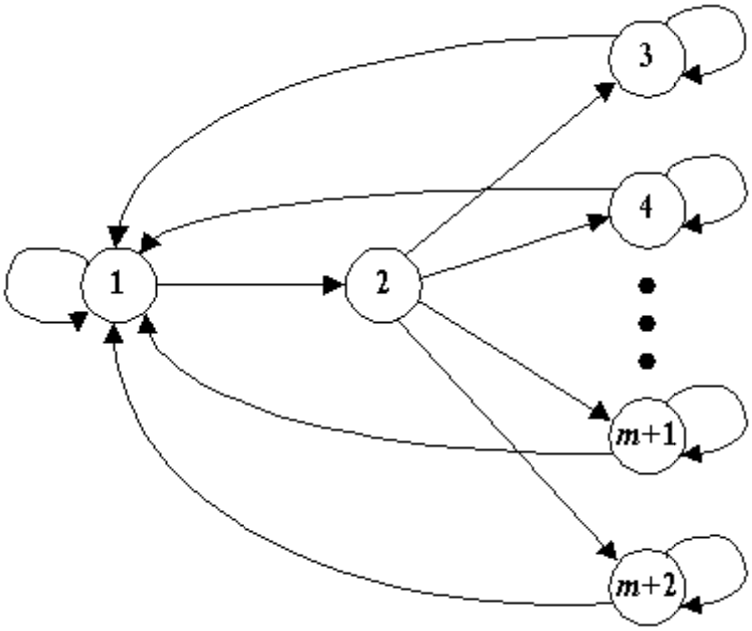


Рис. 4.3. Граф состояний системы адаптивного вычисления ПТФ: 1 – начальное состояние; 2 – выбор вычислительной структуры; 4, ..., m + 1, m + 2 – расчет с использованием 1-й,...,m–1-й или m-й вычислительных структур

Система адаптивного вычисления прямых тригонометрических функций имеет ограниченный ареал использования: ее целесообразно применять в тех случаях, когда условия расчета функций (например, законы изменения аргумента ПТФ) или точность вычислений (следовательно, и разрядность чисел) меняются с течением времени. В случае, если условия расчета остаются постоянными, то более целесообразны другие варианты реализации (например, на основе заказных микросхем). При этом в качестве критериев выбора аппаратного обеспечения могут выступать экономические (себестоимость изделия, общие затраты на обслуживание в течение некоторого периода) и технические (потребляемая мощность и время вычисления ПТФ) характеристики изделия. В первом приближении совокупность этих показателей может быть заменена значениями временных и аппаратных затрат. Таким образом, основными целями разработки математической модели системы адаптивного вычисления ПТФ являются:

- оценка времени расчета функций,
- оценка объема ЗУ конфигураций,

– оценка необходимой логической емкости ДР ПЛИС,

– определение границ применимости подобного устройства [48].

Для решения поставленных задач необходимо знать общее время расчета m значений ПТФ с помощью произвольной конфигурации, формула для которого имеет вид

$$T_{\text{РДПС}} = m(t_{\text{РФ}} + k_{\text{ЗВД}} t_{\text{ЗВД}}) + k_{\text{КОНФ}} t_{\text{КОНФ}} + k_{\text{АДАПТ}} t_{\text{АДАПТ}},$$

где $t_{\text{РФ}}$ – время вычисления функции; $k_{\text{ЗВД}}$ – коэффициент, учитывающий то, что загрузка/выгрузка данных может производиться параллельно с расчетом функций; $t_{\text{ЗВД}}$ – время загрузки/выгрузки данных; $k_{\text{КОНФ}}$ – коэффициент, учитывающий то, что конфигурирование части ПЛИС может производиться параллельно с расчетом функций (для ПЛИС с полной реконфигурацией $k_{\text{жмнФ}} = 1,0$); $t_{\text{КОНФ}}$ – время конфигурирования ДР ПЛИС; $k_{\text{АДАПТ}}$ – коэффициент, учитывающий то, что адаптация может производиться параллельно с расчетом функций ($k_{\text{АДАПТ}}$, $k_{\text{ЗВД}}$, $k_{\text{КОНФ}} \in (0, 1]$); $t_{\text{АДАПТ}}$ – время адаптации к точности расчетов. Значения коэффициентов $k_{\text{ЗВД}}$, $k_{\text{АДАПТ}}$ и $k_{\text{КОНФ}}$ вычисляются, исходя из конкретной архитектуры ДРС [48].

Среднее время расчета функций с помощью системы адаптивного вычисления ПТФ [48] определяется как

$$\bar{t}_{\text{РДПС}} = \frac{m(t_{\text{РФ}} + k_{\text{ЗВД}} t_{\text{ЗВД}}) + k_{\text{КОНФ}} t_{\text{КОНФ}} + k_{\text{АДАПТ}} t_{\text{АДАПТ}}}{m}. \qquad (4.1)$$

Время конфигурирования ДР ПЛИС рассчитывается по формуле [30]

$$t_{\text{КОНФ}} = \frac{n_{\text{lb}} V_{\text{lb}} (1-k)}{f_{\text{КА}}} + t_{\text{ИНИЦ}}, \qquad (4.1.2)$$

где n_{lb} – число логических блоков в ДР ПЛИС, *для ПЛИС* (*с полной реконфигурацией*) или число используемых логических блоков ДР ПЛИС *для ПЛИС* (*с частичной реконфигурацией*); V_{lb} – объем конфигурационных данных одного логического блока; k – коэффициент компрессии конфигурационных данных; $f_{\text{КА}}$ – тактовая частота конфигурационного автомата ДР ПЛИС; $t_{\text{ИНИЦ}}$ – время инициализации внутренних структур ДР ПЛИС [48].

Время загрузки/выгрузки данных зависит от конкретной архитектуры ДРС и не может быть определено в общем виде. Так, для ДРВ время загрузки/выгрузки данных [30] оценивается величиной

$$t_{\text{ЗВД}} = \frac{t_{\text{Ш}} + t_{\text{ОБР}}}{n_{\text{Ш}}} \sum_{\text{j=1}}^{\text{p}} n_{\text{j}}, \tag{4.3}$$

где $t_{\text{Ш}}$ − время цикла шины данных; $t_{\text{ОБР}}$ − время обработки одного слова получаемых/передаваемых данных центральным процессором; $n_{\text{Ш}}$ − разрядность шины данных; n_{j} − разрядность j-го результата или исходного параметра, p – общее число получаемых/передаваемых данных. Если все входные и выходные данные, используемые данной конфигурацией, имеют одинаковую разрядность $n_{\text{ДРС}}$, то выражение (4.3) [48] можно преобразовать к виду

$$t_{\text{ЗВД}} = p\frac{n_{\text{ДРС}}}{n_{\text{Ш}}}(t_{\text{Ш}} + t_{\text{ОБР}}). \tag{4.4}$$

Подставляя (4.2) в (4.1), можно записать [48]:

$$\bar{t}_{\text{РДРС}} = t_{\text{РФ}} + k_{\text{ЗВД}} t_{\text{ЗВД}} + k_{\text{КОНФ}} \frac{n_{\text{lb}} V_{\text{lb}} (1-k)}{f_{\text{КА}} m} + \frac{k_{\text{КОНФ}} t_{\text{ИНИЦ}} + k_{\text{АДАПТ}} t_{\text{АДАПТ}}}{m}. \tag{4.5}$$

Для ДРВ выражение примет вид [48]:

$$\bar{t}_{\text{РДРС}} = t_{\text{РФ}} + k_{\text{ЗВД}} p\frac{n_{\text{ДРС}}}{n_{\text{Ш}}}(t_{\text{Ш}} + t_{\text{ОБР}}) + k_{\text{КОНФ}}\left(\frac{n_{\text{lb}} V_{\text{lb}} (1-k)}{f_{\text{КА}} m} + \frac{t_{\text{ИНИЦ..}}}{m}\right) + \frac{k_{\text{АДАПТ}} t_{\text{АДАПТ}}}{m}. \tag{4.6}$$

Условием применимости системы адаптивного вычисления ПТФ является выполнение системы неравенств [48]:

$$\begin{cases} \bar{t}_{\text{РДРС}} << t_{\text{ПРОГР}}, \\ \bar{t}_{\text{РДРС}} < t_{\text{АПП}} + k_{\text{ЗВАПП}} t_{\text{ЗВАПП}}, \end{cases} \tag{4.7}$$

где $t_{\text{ПРОГР}}$ − время программного расчета ПТФ; $t_{\text{АПП}}$ − время аппаратного расчета ПТФ с использованием одинаковой элементной базы, но без адаптации к точности вычислений; $t_{\text{ЗВАПП}}$ − время загрузки/выгрузки данных для аппаратного расчета ПТФ без адаптации к точности вычислений. Подставляя (4.5) в (4.7), получается система неравенств [48]:

$$\begin{cases} t_{\text{РФ}} + k_{\text{ЗВД}} t_{\text{ЗВД}} + k_{\text{КОНФ}} \dfrac{n_{\text{lb}} V_{\text{lb}} (1-k)}{f_{\text{КА}} m} + \dfrac{k_{\text{КОНФ}} t_{\text{ИНИЦ}} + k_{\text{АДАПТ}} t_{\text{АДАПТ}}}{m} << t_{\text{ПРОГР}} \\ t_{\text{РФ}} + k_{\text{ЗВД}} t_{\text{ЗВД}} + k_{\text{КОНФ}} \dfrac{n_{\text{lb}} V_{\text{lb}} (1-k)}{f_{\text{КА}} m} + \dfrac{k_{\text{КОНФ}} t_{\text{ИНИЦ}} + k_{\text{АДАПТ}} t_{\text{АДАПТ}}}{m} < t_{\text{АПП}} + k_{\text{ЗВАПП}} t_{\text{ЗВАПП}} \end{cases}$$

Если ввести обозначения $\Delta_{\text{ПРОГР}} = t_{\text{ПРОГР}} - t_{\text{РФ}}$, $\Delta_{\text{АПП}} = t_{\text{АПП}} - t_{\text{РФ}}$ и преобразовать оба неравенства последней системы, то условия целесообразности применения системы адаптивного вычисления ПТФ можно переписать в виде [48]

$$\begin{cases} \Delta_{\text{ПРОГР}} \gg k_{\text{КОНФ}} \dfrac{n_{\text{lb}} V_{\text{lb}} (1-k)}{f_{\text{КА}} m} + \dfrac{k_{\text{КОНФ}} t_{\text{ИНИЦ}} + k_{\text{АДАПТ}} t_{\text{АДАПТ}}}{m} + k_{\text{ЗВД}} t_{\text{ЗВД}} \\ \Delta_{\text{АПП}} > k_{\text{КОНФ}} \dfrac{n_{\text{lb}} V_{\text{lb}} (1-k)}{f_{\text{КА}} m} + \dfrac{k_{\text{КОНФ}} t_{\text{ИНИЦ}} + k_{\text{АДАПТ}} t_{\text{АДАПТ}}}{m} + k_{\text{ЗВД}} t_{\text{ЗВД}} - k_{\text{ЗВАПП}} t_{\text{ЗВАПП}} \end{cases} \quad (4.8)$$

Если априори известно, что время программного расчета неприемлемо велико, то система (4.8) преобразуется в неравенство [48]:

$$\Delta_{\text{АПП}} > k_{\text{КОНФ}} \frac{n_{\text{lb}} V_{\text{lb}} (1-k)}{f_{\text{КА}} m} + \frac{k_{\text{КОНФ}} t_{\text{ИНИЦ}} + k_{\text{АДАПТ}} t_{\text{АДАПТ}}}{m} + k_{\text{ЗВД}} t_{\text{ЗВД}} - k_{\text{ЗВАПП}} t_{\text{ЗВАПП}}. \quad (4.9)$$

Очевидно, что для выполнения неравенств (4.8) необходимо [48]:

– сокращать время вычисления функций $t_{\text{РФ}}$;

– конвейеризировать выполнение различных этапов расчета ПТФ (формально это выражается в уменьшении значений коэффициентов $k_{\text{ЗВД}}$, $k_{\text{АДАПТ}}$ и $k_{\text{КОНФ}}$);

– уменьшать время адаптации к точности вычислений $t_{\text{АДАПТ}}$;

– увеличивать число расчетов ПТФ без реконфигурации m;

– сокращать разрядность $n_{\text{ДРС}}$ до минимальной, которая удовлетворяет заданной точности вычислений.

Уменьшение времени расчета ПТФ $t_{\text{РФ}}$ связано как с выбором математического аппарата, так и с выбором структурной организации операционного автомата для каждой конфигурации. Сокращение времени адаптации к точности вычислений $t_{\text{АДАПТ}}$ возможно за счет применения более простого алгоритма адаптации. Увеличение числа операций расчета ПТФ без реконфигурирования возможно при обработке массивов данных и при использовании ПТФ в различных итерационных алгоритмах [30, 31, 48].

В [30, 65] рассмотрены возможности предварительного выбора и загрузки конфигураций, оценена эффективность предлагаемого подхода, скорректированы расчетные выражения. Аналогичные измене-

ния могут быть внесены в (4.5) и (4.9). Кроме того, в [42] произведена оценка объема ЗУ для конфигурационных данных, который вычисляется по формуле

$$V_{\text{ЗУ}} \approx \sum_{l=1}^{s} n_{\text{lb}\,l} V_{\text{lb}} (1 - k_l), \tag{4.10}$$

где s – число конфигураций ДРС; k_l – коэффициент компрессии l-й конфигурационной программы,. Минимальная логическая емкость ДР ПЛИС определяется следующим образом [48]:

$$V_{\text{ПЛИС MIN}} = \max\left\{n_{\text{lb}\,l}\right\}_{l=1}^{s}. \tag{4.11}$$

Если конфигурации используют реконфигурируемые модули памяти, то выражение (4.11) следует переписать в виде

$$\begin{cases} V_{\text{ПЛИС MIN}} = \max\left\{n_{\text{lb}\,l}\right\}_{l=1}^{s}, \\ V_{\text{ПАМЯТЬ MIN}} = \max\left\{V_{\text{ПАМЯТЬ}\,l}\right\}_{l=1}^{s} \end{cases}, \tag{4.12}$$

где $V_{\text{ПАМЯТЬ MIN}}$ – минимально необходимый объем памяти, который должна содержать ПЛИС в виде EAB-модулей; $V_{\text{ПАМЯТЬ}\,l}$ – объем памяти EAB-модулей, который использует l-я конфигурация [48]. При этом следует учитывать, что блоки памяти могут быть организованы на основе логических преобразователей табличного типа [48].

Приведенные выражения позволяют оценить время вычисления ПТФ с помощью произвольной конфигурации, рассчитать объем ЗУ конфигураций, логическую емкость ДР ПЛИС, сравнить различные варианты реализации вычислений ПТФ и на основании этих данных произвести адекватный выбор элементной базы для устройств расчета указанных функций [48].

4.2. Устройство адаптивного сжатия данных

Развитие телекоммуникационных систем привело к значительному увеличению объемов передаваемых данных, поэтому пропускной способности многих каналов связи не всегда достаточно, чтобы обеспечить передачу требуемого трафика за приемлемое время. Кроме того, в настоящий момент получили широкое распространение системы архивации данных различных типов (например, в системах мониторинга), ко-

торые требуют значительных емкостей запоминающих устройств. В [8, 36] показано, что адаптивное сжатие данных (АСД) является эффективным средством сокращения объема трафика, позволяя при этом использовать различные алгоритмы компрессии в зависимости от условий применения и информации, содержащейся в данных. Вместе с тем, немаловажным фактором при проектировании устройства адаптивного сжатия данных (УАСД) является современная элементная база, накладывающая ограничения на сложность алгоритмов адаптации [62]. По этим причинам разработку УАСД можно отнести к числу самых сложных и актуальных научно-технических проблем [42].

Наиболее трудоемким и до настоящего момента не формализованным этапом АСД является выбор алгоритма компрессии данных. Для последующей технической реализации данного этапа его необходимо представить в виде особого алгоритма, который назовем алгоритмом адаптации. Разработку подобного алгоритма целесообразно начать с формализации критериев выбора. Вполне очевидно, что основным критерием выбора алгоритма компрессии является степень компрессии $K_{\text{компр}}$ для конкретного типа данных, то есть количественный критерий, показывающий степень сокращения первоначального объема данных в результате компрессии:

$$K_{\text{компр}} = \frac{V}{V_{\text{сж}}},$$

где V – объем данных до компрессии, бит; $V_{\text{сж}}$ – объем данных после компрессии, бит [42].

В формализованном виде задачу, решаемую алгоритмом адаптации при сжатии данных, можно представить в терминах теории множеств: необходимо найти такое преобразование f_{a}, которое позволяет отобразить множество алгоритмов компрессии C на множество различных типов данных D [42]: $f_a: C \to D$.

Сложность данной задачи, помимо всего прочего, заключается в том, что данное отображение является сюръективным, но не является инъекцией, то есть не обладает свойством взаимной однозначности. Следовательно, некоторому произвольному элементу c_{k} множества C может соответствовать не одно, а несколько элементов из множества D [42], то есть $f_a(-_k) = f_a(c_m) = ... = f_a(c_n)$.

Более того, данное утверждение справедливо и для некоторого преобразования f_a^{-1}, которое позволяет отобразить множество различных типов данных D на множество алгоритмов компрессии C: $f_a^{-1}: D \to C$, а именно: некоторому произвольному элементу d_k множества D может соответствовать не один, а несколько элементов из множества C, то есть

$$f_a^{-1}(d_k) = f_a^{-1}(d_m) = ... = f_a^{-1}(d_n).$$

Данные утверждения имеют следующий технический смысл: один и тот же алгоритм сжатия может быть использован для компрессии различных типов данных, а также один и тот же тип данных может быть сжат с использованием различных алгоритмов компрессии. Очевидно, что с технической точки зрения, наиболее предпочтительным является такой алгоритм адаптации, который бы взаимно однозначно отображал множество алгоритмов компрессии C на множество различных типов данных D, то есть чтобы отображение было биективным. С этой целью в алгоритме адаптации необходимо использовать дополнительные критерии, которые позволят устранить подобную неоднозначность. Такие критерии можно разделить на две основные группы [42]:

– ***не зависящие от реализации***: *степень асимметрии алгоритма сжатия* $K_{\text{асим}} = \dfrac{T_{\text{компр}}}{T_{\text{декомпр}}}$ – это функция двух других критериев – времени компрессии данных $T_{\text{компр}}$ и времени декомпрессии данных $T_{\text{декомпр}}$, $K_{\text{асим}} = f\left(T_{\text{компр}}, T_{\text{декомпр}}\right)$, однако, несмотря на это, критерий инвариантен к технической реализации; *уровень допустимых потерь информации* (для сжатия с потерями), который формально может быть оценен с помощью множества величин $\{L\}$ (например, максимальная и среднеквадратическая ошибки восстановления, отношение сигнал/шум); *информация, содержащаяся в данных*, что можно охарактеризовать множеством величин $\{I\}$, например, тип данных (видео-, аудиоданные, статические изображения, трехмерные модели и т.п.), значения градиента цветности в различных точках изображения, максимальное значение звукового давления;

– ***зависящие от реализации***: время компрессии данных $T_{\text{компр}}$, время декомпрессии данных $T_{\text{декомпр}}$, *параметры канала связи* $\{Q\}$, включающие в себя скорость передачи данных R, характеристики помехоустойчивости и т.д. [42].

Функции алгоритма адаптации заключаются не только в выборе самого N-го алгоритма компрессии, но также и в расчете набора параметров $\{P\}_N$ соответствующего алгоритма компрессии (например, значений по шкале «качество-размер» для алгоритмов JPEG и JPEG2000). Формально данную процедуру можно записать в виде уравнения [42]:

$$\{P\}_N = f(K_{\text{компр}}, T_{\text{компр}}, T_{\text{декомпр}}, \{L\}, \{I\}, \{Q\}).$$

Учитывая сложность задач АСД, при проектировании УАСД целесообразно использовать методы системного анализа. В этой связи, как следует из литературы [19, 24, 26, 54], необходимо произвести декомпозицию УАСД, разработав его иерархическую структуру. Наиболее естественным подходом к декомпозиции объекта исследования является декомпозиция алгоритмического обеспечения (АО). Иерархическая структура АО УАСД представлена на рис. 4.4.

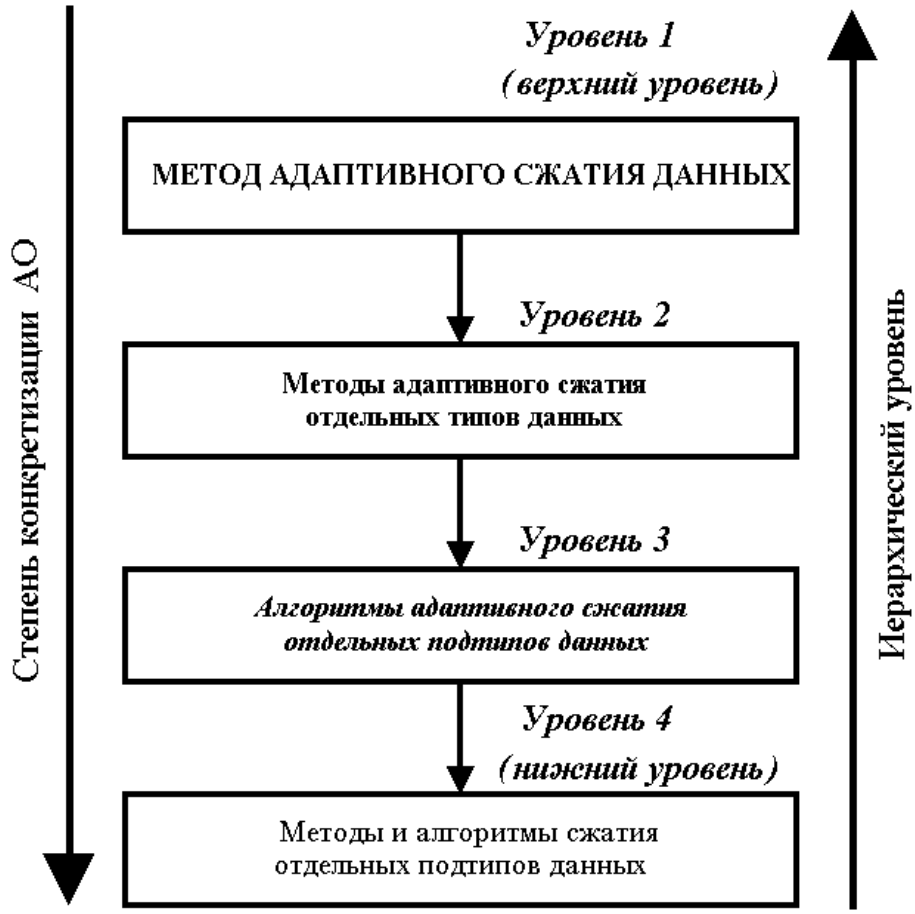


Рис. 4.4. Иерархическая структура алгоритмического обеспечения УАСД

Как следует из данной блок-схемы, декомпозиция АО УАСД производилась с учетом двух основных критериев [42]:

– типа обрабатываемых данных,

– степени общности методов компрессии/декомпрессии.

Иерархическая структура АО УАСД полностью соответствует технологии объектно-ориентированного программирования (ООП): создается некоторый класс «общий предок», реализующий самые общие методы и свойства, которые характерны для адаптивной компрессии/декомпрессии данных и которые затем наследуются классами более низкого иерархического уровня. Самый нижний уровень иерархии соответствует методам адаптивного сжатия отдельных типов данных. Таким образом, по мере движения от класса «общего предка» к классам более низкого иерархического уровня происходит наполнение последних все более конкретными методами и свойствами, которые определяют особенности каждого метода и алгоритма компрессии. Класс самого низкого уровня обеспечивает компрессию/декомпрессию данных по конкретному алгоритму (например, алгоритм Хаффмана, RLE, LZW, арифметическое кодирование, lossless JPEG – при сжатии без потерь; JPEG, JPEG2000, MPEG-1, -2, -4, H.263, H.264 – при компрессии с потерями) [42].

Следующий этап формализации задач АСД −разработка алгоритма адаптации. Обобщенный алгоритм адаптации (рис. 4.5) отражает основные этапы выбора алгоритма, однако в каждом конкретном случае алгоритм адаптации разрабатывается отдельно с учетом частных условий и ограничений [42].

Рассмотрим каждый из этапов обобщенного алгоритма адаптации. Анализ содержания данных и его формальное выражение в виде некоторого множества величин $\{I\}$ составляет отдельную научную задачу и является предметом исследований такой области науки, как теория распознавания образов [42]. В рамках данной теории разработано довольно большое количество методов, классификация и обобщение которых для различных типов данных проведены, например, в [69].

Расчет зависимых критериев ($K_{\text{компр}}$, $T_{\text{компр}}$, $T_{\text{декомпр}}$) специфичен для каждого алгоритма компрессии, а расчет критериев $T_{\text{компр}}$ и $T_{\text{декомпр}}$, отражающих скорость выполнения операции компрессии, связан с анализом особенностей реализации, что требует использования

моделей вычислительных средств, которые применялись при реализации УАСД. Таким образом, построение моделей, инвариантных к алгоритмам компрессии, для двух вышеописанных этапов не представляется возможным [42].

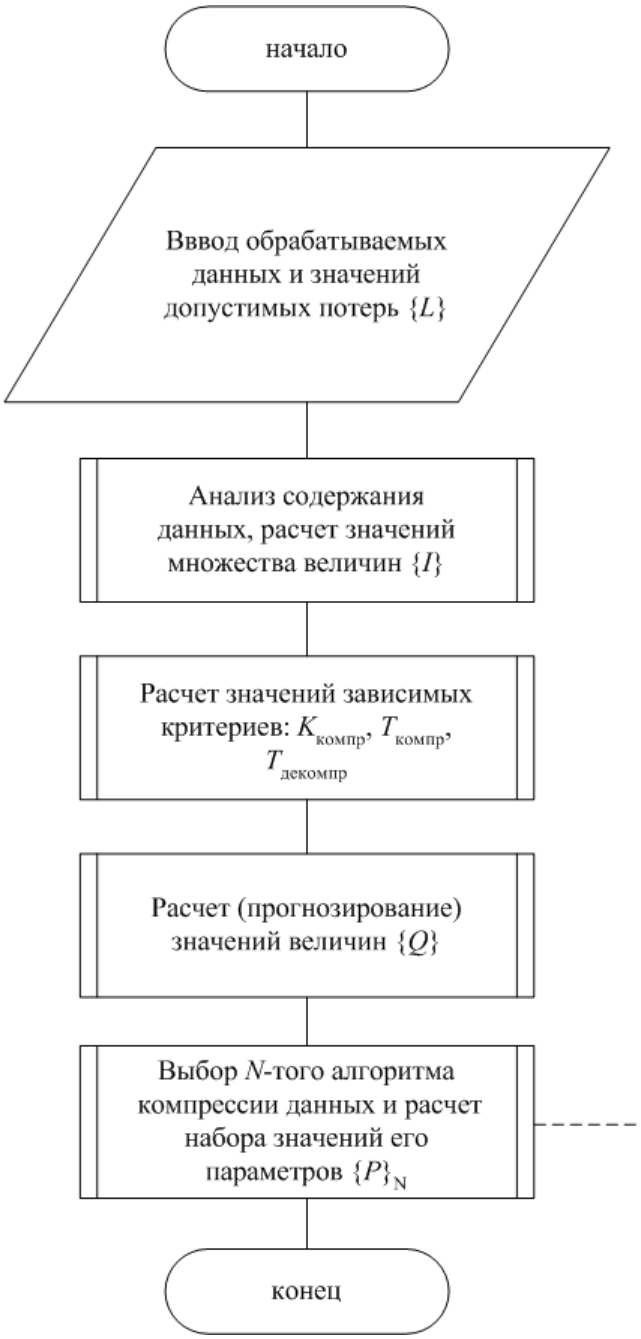


Рис. 4.5. Обобщенный алгоритм адаптации

Одной из важных задач при выборе алгоритма компрессии мультимедийных данных является расчет (оценка) значений величин $\{Q\}$ (в первую очередь, значения скорости передачи данных R в канале связи).

Этот критерий во многом определяет выбор алгоритма сжатия, более того – необходимость компрессии вообще, поскольку для каждой задачи существует некоторое пороговое значение скорости передачи данных *R*. Превышение этого порогового значения приводит к нецелесообразности сжатия данных ввиду избыточных временных затрат. В то же время данный критерий является в рамках данной задачи независимым, поэтому для него возможно построение модели, инвариантной по отношению к задаче АСД и отражающей процессы, происходящие в канале связи [42].

Во многих случаях при проектировании электронных средств оказывается целесообразнее использовать не алгоритмическое, а структурное описание системы. По этой причине была разработана структурная схема АСД (рис. 4.6), которая во многом является отражением процессов, описываемых алгоритмом адаптации. Ее специфичность заключается в том, что она определяет все этапы АСД – от получения исходных данных до формирования потока сжатых данных [42].

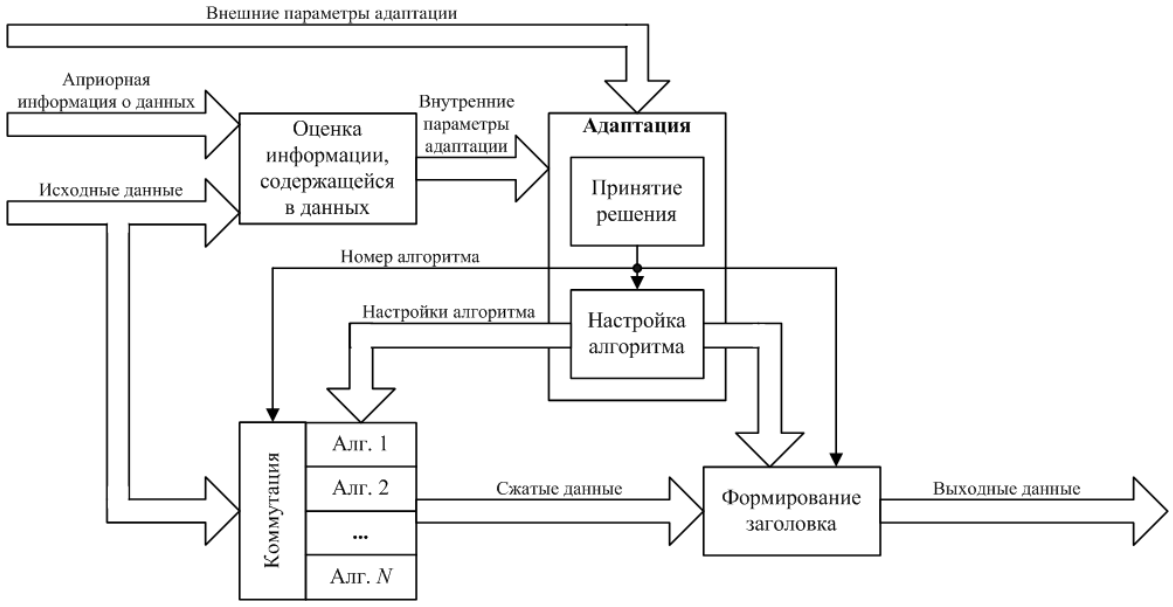


Рис. 4.6. Структурная схема АСД

В связи с тем, что одни и те же методы и алгоритмы могут применяться для компрессии различных типов данных и наоборот – для сжатия данных одного и того же типа можно использовать различные методы и алгоритмы, то отдельной научной задачей является разработка алгоритма адаптации УАСД, используемого для выбора оптимального алгоритма компрессии на основе параметров передаваемых

данных. Такая процедура требует предварительного анализа данных, включающего в себя их декомпозицию до элементарных структур (текст, звук, видео, изображения и т.п.), если данные являются составными (например, doc-файлы текстового процессора Microsoft Word, базы данных и т.д.). На основании всего этого можно предложить обобщенные алгоритмы функционирования УАСД в режимах компрессии (рис. 4.7) и декомпрессии (рис. 4.8) [42].

Из рис. 4.7 и 4.8 видно, что при передаче простых типов данных УАСД будет обладать свойством асимметрии: компрессия данных будет производиться дольше, чем декомпрессия, т.к. в последнем случае нет необходимости анализировать получаемые данные и выбирать метод декомпрессии (данные об использованном алгоритме сжатия будут инкапсулированы в заголовок файла). Однако в случае передачи составных данных в режиме декомпрессии требуется производить их восстановление (соотношение времени декомпозиции и восстановления во многом зависит от типов данных), что позволит компенсировать асимметрию и перейти в квазисимметричный режим работы УАСД. Формально условие, при котором устройство находится в квазисимметричном режиме работы (в режиме, близком к симметричному), можно определить как

$$T_{\text{компр}} >> T_{\text{анализ}} + T_{\text{адапт}},$$

где $T_{\text{компр}}$ − время компрессии данных одного типа, $T_{\text{анализ}}$ − время анализа данных, $T_{\text{адапт}}$ − время выбора алгоритма сжатия. Технически квазисимметричному режиму соответствует случай, когда длительное время осуществляется передача данных одного и того же типа [36].

В обобщенном алгоритме функционирования УАСД требуется определять необходимость компрессии данных. Очевидно, что данные не следует сжимать в случае, когда временные затраты на компрессию/декомпрессию будут больше, чем сокращение временных затрат на передачу данных (например, сжатие уже заархивированных данных – файлов zip, rar, arj, jpg и т.д.). Поэтому условие эффективности УАСД при передаче данных в канале линии связи можно определить следующим образом: пусть требуется передать V бит данных по каналу связи с пропускной способностью C бит/с. В этом случае время передачи без использования УАСД рассчитывается по формуле [36]

$$T_{\text{перед}} = \frac{V}{C}. \qquad (4.13)$$

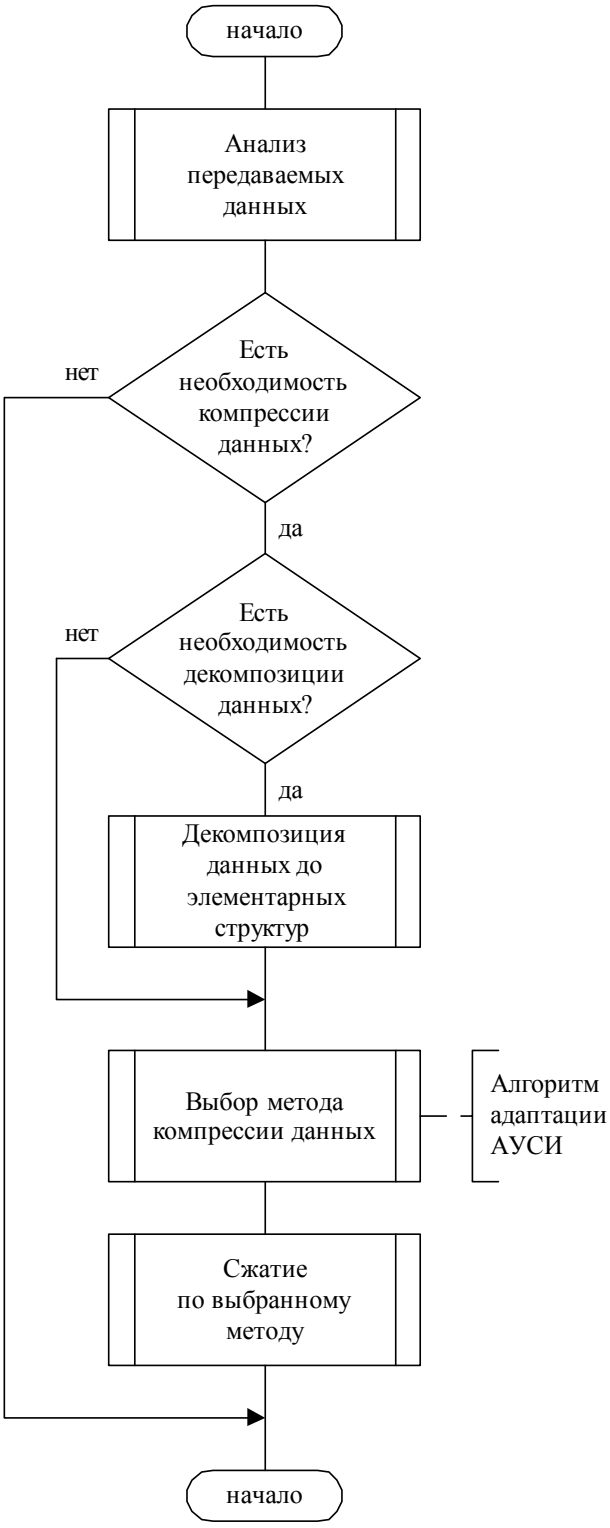


Рис. 4.7. Обобщенный алгоритм функционирования УАСД в режиме компрессии

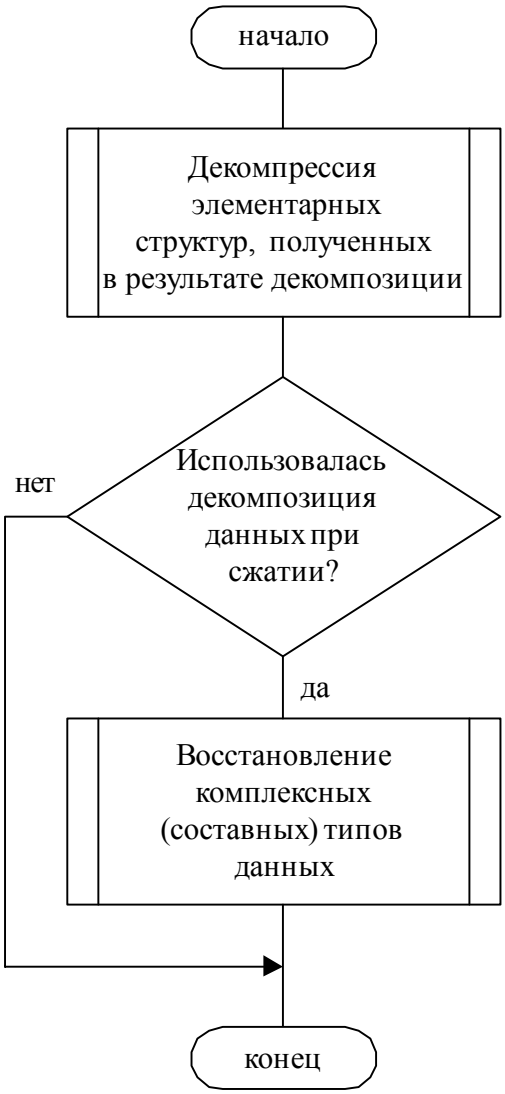


Рис. 4.8. Обобщенный алгоритм функционирования УАСД в режиме декомпрессии

Если произвести компрессию данных по некоторому алгоритму, то можно записать [36], что

$$V = V_{\text{сж}} + \Delta V, \tag{4.14}$$

где $V_{\text{сж}}$ − объем данных после сжатия, ΔV − уменьшение объема данных в результате сжатия. Подставляя (4.14) в (4.13), согласно [36] получается

$$T_{\text{пер}} = \frac{V_{\text{сж}} + \Delta V}{C} = \frac{V_{\text{сж}}}{C} + \frac{\Delta V}{C} = T_{\text{пер}}^{\text{сж}} + \Delta T, \tag{4.15}$$

где $T_{\text{пер}}^{\text{сж}}$ − время передачи сжатых данных, ΔT − разница во времени передачи сжатых и несжатых данных. Кроме того, следует учитывать необходимость дополнительной обработки данных (например, разбиение на кадры в локальных сетях) [36]:

$$T_{\text{обр}}^{\text{сж}} = T_{\text{обр}} + T_{\text{к-д}}, \tag{4.16}$$

где $T_{\text{обр}}$ − время обработки несжатых данных, $T_{\text{к-д}}$ − суммарное время компрессии и декомпрессии данных, $T_{\text{обр}}^{\text{сж}}$ − общее время обработки сжатых данных [36].

Если пренебречь разницей во времени между предварительной обработкой исходных и сжатых данных в виду ее малости ($T_{\text{обр}}^{\text{исх}} = T_{\text{обр}}^{\text{сж}} = T_{\text{обр}}$), то полное время передачи и предварительной обработки данных вычисляется по формуле [36]

$$T_{\text{полн}} = T_{\text{пер}} + T_{\text{обр}}. \tag{4.17}$$

Эффективность компрессии данных по некоторому произвольному алгоритму будет определяться условием [36]

$$T_{\text{полн}}^{\text{сж}} < T_{\text{полн}}. \tag{4.18}$$

Подставляя в (4.18) формулы (4.16) и (4.17), можно записать [36]:

$$T_{\text{пер}} - \Delta T + T_{\text{обр}} + T_{\text{к-д}} < T_{\text{пер}} + T_{\text{обр}}. \tag{4.19}$$

Приведя подобные в неравенстве (4.19) и преобразовав его, получается выражение [36]

$$\Delta T > T_{\text{к-д}}. \tag{4.20}$$

Так как $\Delta T = \frac{\Delta V}{C}$, то (4.20) примет вид [36]

$$\frac{\Delta V}{C} > T_{\text{к-д}}. \tag{4.21}$$

При передаче данных одного типа целесообразность применения УАСД по сравнению с устройством, использующим только один алгоритм сжатия, определяется системой неравенств [36]:

$$\begin{cases} \dfrac{\Delta V}{C} > T_{\text{к–д}}^{\text{опт}} + T_{\text{анализ}} + T_{\text{адапт}} + T_{\text{дек}} + T_{\text{восст}}, \\ T_{\text{полн}}^{\text{сж.пр.}} > T_{\text{полн}}^{\text{сж.опт.}} \end{cases} \quad (4.22)$$

где $T_{\text{к-д}}^{\text{опт}}$ − время компрессии и декомпрессии по оптимальному в данной ситуации алгоритму сжатия (в случае составных данных − это усредненный показатель); $T_{\text{дек}}$ − время декомпозиции составных данных; $T_{\text{восст}}$ − время восстановления составных данных; $T_{\text{полн}}^{\text{сж.пр.}}$ − полное время передачи и обработки для данных, сжатых произвольным алгоритмом (например, стандартным для модема) без учета специфики данных; $T_{\text{полн}}^{\text{сж.опт.}}$ − полное время передачи и обработки для данных, сжатых УАСД. Время передачи и полной обработки данных, сжатых УАСД, рассчитывается по формуле [36]:

$$T_{\text{полн}}^{\text{сж.опт.}} = T_{\text{пер}} - \Delta T_{\text{опт}} + T_{\text{анализ}} + T_{\text{адапт}} + T_{\text{дек}} + T_{\text{восст}} + T_{\text{обр}} + T_{\text{к–д}}^{\text{опт}}, \quad (4.23)$$

где $\Delta T_{\text{опт}}$ − разница во времени передачи исходных и сжатых данных при использовании оптимального алгоритма [36].

Второе уравнение системы (4.22) с учетом (4.23), (4.16) и (4.17) может быть преобразовано к виду [36]

$$T_{\text{пер}} - \Delta T_{\text{пр.}} + T_{\text{обр.}} + T_{\text{к-д}}^{\text{пр.}} > T_{\text{пер}} - \Delta T_{\text{опт}} + T_{\text{анализ}} + T_{\text{адапт}} + T_{\text{дек}} + T_{\text{восст}} + T_{\text{обр}} + T_{\text{к–д}}^{\text{опт}}, \quad (4.24)$$

где $\Delta T_{\text{пр.}}$ − разница во времени передачи исходных и сжатых данных при использовании произвольного алгоритма, применяемого в модеме. Приводя подобные в (4.24), можно записать [36]:

$$T_{\text{к-д}}^{\text{пр.}} - \Delta T_{\text{пр.}} > T_{\text{анализ}} + T_{\text{адапт}} + T_{\text{дек}} + T_{\text{восст}} + T_{\text{к–д}}^{\text{опт}} - \Delta T_{\text{опт}}. \quad (4.25)$$

Очевидно [36], что

$$\Delta T_{\text{опт}} = \Delta T_{\text{пр.}} + \Delta T_{\text{УАСД}}, \quad (4.26)$$

где $\Delta T_{\text{УАСД}}$ − время, на которое сокращается передача данных в результате использования УАСД по сравнению с произвольным алгоритмом сжатия, применяемом в модеме.

При этом [36]

$$\Delta T_{\text{УАСД}} = \frac{\Delta V_{УАСД}}{C}, \quad (4.27)$$

где $\Delta V_{\text{УАСД}}$ − уменьшение объема данных в результате применения УАСД по сравнению с произвольным алгоритмом, используемым в модеме.

С учетом (4.26) и (4.27) неравенство (4.25) [36] можно записать в виде

$$\frac{\Delta V_{\text{УАСД}}}{C} > T_{\text{анализ}} + T_{\text{адапт}} + T_{\text{дек}} + T_{\text{восст}} + T_{\text{к–д}}^{\text{опт}} - T_{\text{к–д}}^{\text{пр}}. \qquad (4.28)$$

Подставляя (4.28) в систему (4.22) вместо второго уравнения, получается, согласно [36]

$$\begin{cases} \dfrac{\Delta V}{C} > T_{\text{к-д}}^{\text{опт}} + T_{\text{анализ}} + T_{\text{адапт}} + T_{\text{дек}} + T_{\text{восст}}, \\ \dfrac{\Delta V_{\text{УАСД}}}{C} > T_{\text{анализ}} + T_{\text{адапт}} + T_{\text{дек}} + T_{\text{восст}} + T_{\text{к-д}}^{\text{опт}} - T_{\text{к-д}}^{\text{пр}}. \end{cases} \qquad (4.29)$$

Система (4.29) может быть использована для определения целесообразности применения УАСД относительно некоторого произвольного алгоритма сжатия, используемого в модеме [36].

УАСД может быть реализовано как на компьютерах различной платформы (программно или аппаратно-программно), так и в виде отдельного устройства (аппаратно или аппаратно-программно). При этом следует учесть некоторые особенности УАСД [36]:

− в энергонезависимой памяти устройства должно храниться достаточно большое число описаний алгоритмов компрессии/декомпрессии (в виде программ или файлов прошивок ПЛИС), однако в каждый момент работы устройства используется только одно из них;

− должна существовать возможность удаления, включения и замены описаний алгоритмов без замены аппаратного обеспечения;

− замена одного описания на другое должна производиться достаточно быстро, чтобы не снизить общую производительность устройства;

− должна существовать возможность коррекции каждого этапа компрессии/декомпрессии (например, в случае изменения алгоритма адаптации, процедуры анализа передаваемых данных и т.д.) без замены аппаратного обеспечения;

− время выполнения операций должно быть минимальным для уменьшения значений правой части неравенств системы (4.29).

На основании перечисленного можно сделать вывод о том, что оптимальным средством реализации УАСД являются динамически реконфигурируемые системы обработки информации. Это обусловлено тем, что такие системы ориентированы на многократное использование одного и того же аппаратного обеспечения для решения различных задач, а также сочетают в себе достоинства аппаратной (высокое быстродействие) и программной (универсальность и гибкость) реализаций, позволяя выполнить требования, изложенные выше [36].

Первый этап реализации УАСД на базе ДРС − разработка его структурной схемы. Данный этап проектирования крайне важен, поскольку определяет основные функциональные технические характеристики будущего устройства. Один из вариантов структурной схемы УАСД, показанный на рис. 4.9, включает блоки, реализуемые как на базе ДРС, которая может иметь различные архитектуры, так и на базе традиционных процессоров.

Рис. 4.9. Структурная схема УАСД

Таким образом, использование динамически реконфигурируемых ПЛИС в рамках ДРС совмещено с применением проблемно-ориентированных (для задач цифровой обработки сигналов, к которым относится и компрессия данных, это – цифровые сигнальные процессоры) и универсальных (например, для формирования пакетов в соответствии со стеком протоколов RTSP/RTP/RTCP) процессоров. Применение подобных вариантов реализации позволит обеспечить производительность, достаточную для обработки данных в режиме реального времени, а также избежать значительных аппаратных и материальных затрат, связанных с изготовлением заказных микросхем для каждого алгоритма компрессии.

Вопросы и задания для самоподготовки

1. Какие примеры применения ДРС, кроме тех, которые были рассмотрены выше, Вы можете привести? Почему в этих случаях целесообразно использование ДРС?

2. Почему задачу вычисления математических функций можно использовать для сравнительной оценки производительности ДРС и традиционных вычислителей? Приведите примеры приложений, в которых наиболее интенсивно применяется расчет математических функций. В каких областях науки и техники используются ПТФ? С чем это связано?

3. В каких случаях требуется менять точность расчета математических функций в пределах одного вычислителя? Какие параметры вычислителя должны быть изменены в таком случае? Обоснуйте Ваше мнение математически.

4. Из каких структурных элементов состоит система адаптивного вычисления ПТФ? Какие задачи выполняют эти элементы? Чем обусловлено такое разделение? Нарисуйте и опишите структурную схему системы адаптивного вычисления ПТФ и граф её состояний.

5. Что является целью математического моделирования системы адаптивного вычисления ПТФ? Приведите основные формулы, которые используются для оценки значений основных технических характеристик системы.

6. Какие ДРС и почему целесообразно применять для реализации системы адаптивного вычисления математических функций?

7. Что такое устройство адаптивного сжатия данных? Какие задачи оно решает? Где оно может применяться? Перечислите названия основных параметров, определяющих технические характеристики УАСД.

8. Какие методы и алгоритмы сжатия данных Вам известны? Где они применяются? Чем они отличаются друг от друга?

9. Из каких структурных элементов состоит УАСД? Какие задачи выполняют эти элементы? Чем обусловлено такое разделение? Нарисуйте и опишите структурную схему адаптивного сжатия данных и блок-схему обобщенного алгоритма адаптации, применяемого в УАСД.

10. Нарисуйте и опишите блок-схему алгоритма функционирования УАСД в режиме компрессии и режиме декомпрессии.

11. Приведите основные математические выражения, используемые при математическом моделировании УАСД.

12. Сравните основные варианты реализации УАСД, используя в качестве критериев оценки комплексный критерий «3P’s» и себестоимость системы. Какой вариант является наиболее предпочтительным? Обоснуйте Ваше мнение.

13. Нарисуйте и опишите структурную схему УАСД. Каким образом можно было бы модифицировать данную схему, чтобы оптимизировать ее с целью минимизации времени обработки или потребляемой мощности? Аргументируйте Ваш ответ.

14. Какие ДРС и почему целесообразно применять для реализации УАСД?

ЗАКЛЮЧЕНИЕ

Настоящее учебное пособие посвящено одному из наиболее перспективных направлений развития современных электронно-вычислительных средств – динамически реконфигурируемым системам обработки информации, которые, как было показано, представляют собой вычислительные устройства, базирующиеся на принципах, отличных от тех, что были предложены Джоном фон Нейманом. Более того, традиционные процессоры являются частным случаем ДРС. Одно из ключевых преимуществ рассматриваемых систем состоит в том, что они инвариантны к используемой элементной базе, то есть принцип, заложенный в ДРС, универсален и может быть применен как к уже существующему аппаратному обеспечению (например, динамически реконфигурируемые ПЛИС и оптические устройства), так и к перспективному (в частности, нет принципиальных ограничений для создания в будущем динамически реконфигурируемых квантовых компьютеров).

Следует отметить, что проектирование ДРС требует большого объема теоретических знаний и серьезной практической подготовки как в области схемотехники и конструирования электронно-вычислительных средств, так и в области программирования. В этой связи целесообразно перечислить области знаний, которые необходимы при проектировании сложных динамически реконфигурируемых систем обработки информации и требуют отдельного глубокого изучения:

− цифровая схемотехника, включая знания в области современной элементной базы и реализации локальных шин передачи цифровых данных (PCI-Express, SPI, I^2C и т.д.);

− автоматизированное проектирование устройств на базе ПЛИС (прежде всего, знания языков описания аппаратурных средств – Hardware Description Languages, в том числе по стандартным языкам – VHDL и Verilog);

− разработка системного и прикладного программного обеспечения для встраиваемых систем, в частности, знания языков низкого (Ассемблер) и высокого (C/C++) уровней;

− операционные системы, включая операционные системы реального времени (например, QNX, LynxOS, VxWorks).

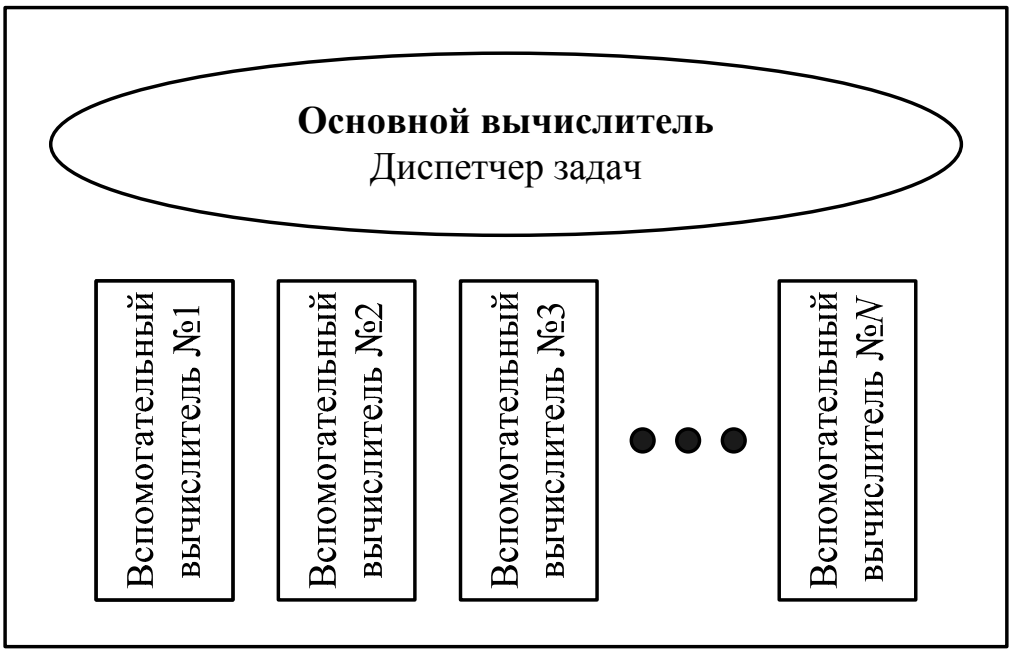
В рамках данной работы рассмотрены теоретические основы, позволяющие студенту понять, что представляют собой ДРС, как они устроены, и каким образом функционируют. В учебном пособии содержится минимально необходимый объем информации, который требуется для проектирования ДРС.

Знания, полученные студентами после ознакомления с настоящим учебным пособием, могут быть использованы ими при курсовом и дипломном проектировании.

ПРИЛОЖЕНИЕ

Пояснения к классификации, характеризующей взаимодействие вычислительных модулей

Централизованные системы обработки информации, состоящие из многих вычислительных модулей и ориентированные на выполнение не одной, а большого числа различных задач, как правило, управляются операционными системами. Их основной элемент − диспетчер, позволяющий распределять задачи между вычислителями (рисунок). Очевидно, что те модули, на базе которых решается указанная задача распределения вычислительных ресурсов, являются основными, так как их отказ повлечет за собой выход из строя системы в целом. Следует подчеркнуть, что каждый вспомогательный вычислитель может или быть простым (например, заказной микросхемой, ориентированной на выполнение заранее определенных задач с жестко фиксированным их числом), или состоять из нескольких более простых вычислителей, которые в свою очередь снова могут быть разделены на основные и вспомогательные. Таким образом, выстраивается иерархия вычислителей в рамках данной классификации. Местоположение в ней определяется типом вычислителя (основной или вспомогательный) и уровнем декомпозиции системы.



Распределение задач в централизованных системах обработки информации, состоящих из множества вычислителей

Математически задачу диспетчеризации можно сформулировать следующим образом: пусть система обработки информации, которая должна обработать параллельно (или квазипараллельно – для одномодульных систем) M задач $G = \{g_j\}_{j=0}^{M-1}$, состоит из N вычислительных модулей $C = \{c_i\}_{i=0}^{N-1}$. В этом случае задача диспетчеризации может заключаться в таком распределении решаемых задач по вычислительным модулям $c_i \mapsto G_i = \{g_{il}\}_{l=0}^{m_i-1}$, где G_i и m_i ($\sum_{i=0}^{N-1} m_i = M$) – набор задач, выполняемых на базе вычислительного модуля c_i, и их количество, соответственно, чтобы было справедливо одно из следующих условий в зависимости от приоритетов, которые стоят перед операционной системой:

1) $T_{g_j} \le T_{g_j}^{\text{thr}}$ для $j = \overline{0, M-1}$;

2) $\max\limits_{i=\overline{1,N-1}} \{T(G_i)\} \to \min$;

где T_{g_j} – время выполнения задачи g_j; $T_{g_j}^{\text{thr}}$ – максимально допустимое время выполнения задачи g_j; $T(G_i)$ – время выполнения набора задач на базе вычислительного модуля c_i. Очевидно, что найти решение для поставленной задачи в общем виде достаточно сложно, поэтому, как правило, можно ограничиться частными решениями, дающими на практике приемлемые результаты.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. *Айлиф, Дж*. Принципы построения базовой машины / Дж. Айлиф; пер. с англ. – М.: Мир, 1973. – 120 с.

2. *Ахо, А.В.* Компиляторы: принципы, технологии и инструменты / А.В. Ахо, Р. Сети, Д.Д. Ульман; пер. с англ. – М.: Вильямс, 2003. – 768 с. – ISBN 5-8459-0189-8.

3. *Байков, В.Д.* Решение траекторных задач в микропроцессорных системах ЧПУ/ В.Д. Байков, С.Н. Вашкевич; под ред. В.Б. Смолова. – Л.: Машиностроение. Ленингр. отд-ние, 1986. – 106 с.

4. *Они же.* Средства реализации алгоритмов интерполяции в современных системах ЧПУ станками / В.Д. Байков, С.Н. Вашкевич // Изв. ЛЭТИ. – 1980. – № 278. – С. 44 – 49.

5. *Байков, В.Д.* Вычисление элементарных функций в ЭКВМ / В.Д. Байков, С.А. Селютин. – М.: Радио и связь, 1982. – 64 с.

6. Байков, В.Д. Аппаратурная реализация элементарных функций в ЭЦВМ / В.Д. Байков, В.Б. Смолов. – Л.: Изд-во ЛГУ, 1975. – 96 с.

7. *Они же.* Специализированные процессоры: итерационные алгоритмы и структуры / В.Д. Байков, В.Б. Смолов. – М.: Радио и связь, 1985. – 288 с.

8. *Барашков, А.В.* Адаптивное сжатие данных / А.В. Барашков, М.В. Руфицкий, А.К. Филиппов // Перспективные технологии в средствах передачи информации : материалы 6 междунар. науч.-техн. конф. – Владимир: РОСТ, 2005. – С. 224 – 226.

9. *Беркс, А.* Предварительное рассмотрение логической конструкции электронного вычислительного устройства / А. Беркс, Г. Голдстейн, Дж. Нейман // Кибернетический сборник. – 1964. – № 9 – С. 7 – 67.

10. *Бернюков, А.К.* Применение динамически реконфигурируемых систем в задачах цифровой обработки биоэлектрических сигналов / А.К. Бернюков, А.К. Филиппов // Физика и радиоэлектроника в медицине и экологии : материалы 8 междунар. науч.-техн. конф. – В 2 кн. Кн. 1. – Владимир: Тип. УВД Владим. обл., 2008. – С. 220 – 224.

11. *Бернюков, А.К.* Реализация алгоритмов функционально-адаптивной обработки сигналов для авиационных бортовых систем на основе высоконадежных динамически реконфигурируемых вычислителей / А.К. Бернюков, А.К. Филиппов // Труды третьей международной научно-технической конференции «Акустооптические и радиолокационные методы измерений и обработки информации». – Владимир: РОСТ, 2009. – С. 28 – 32.

12. *Бернюков, А.К.* Специализированные вычислители для цифровой обработки биоэлектрических сигналов / А.К. Бернюков, А.К. Филиппов // Биомедицинская радиоэлектроника. – 2008. – № 6. – С. 72 – 78.

13. *Они же.* Специализированные вычислители на базе динамически реконфигурируемых ПЛИС для использования в цифровых телекоммуникационных системах / А.К. Бернюков, А.К. Филиппов // Материалы седьмой международной научно-технической конференции «Перспективные технологии в средствах передачи информации – ПТСПИ 2007». – Владимир: Изд-во Владимир. гос. ун-та, 2007. – С. 167 – 171.

14. *Ватолин, Д.* Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов. – М.: ДИАЛОГ-МИФИ, 2002. – 384 с.

15. *Вашкевич, С.Н.* Алгоритмы формообразования для микропроцессорных систем числового программного управления / С.Н. Вашкевич // Автоматизация процессов обработки первичной информации : межвуз. сб. науч. тр. – 1982. – № 8. – С. 28 – 31.

16. *Вентцель, Е.С.* Теория вероятностей / Е.С. Вентцель. – М.: Наука, 1969. – 576 с.

17. *Глушков, В.М.* Теория автоматов и формальные преобразования микропрограмм / В.М. Глушков // Кибернетика. –1965. – № 5. – С. 1 – 9.

18. *Горбатов, В.А.* Семантическая теория проектирования автоматов / В.А. Горбатов. – М.: Энергия, 1979. – 264 с.

19. *Губанов, В.А.* Введение в системный анализ / В.А. Губанов, В.В. Захаров, А.Н. Коваленко. – Л.: Изд-во ЛГУ, 1988. – 231 с.

20. *Ивленков, М.Ю.* Аппаратно-программный комплекс для исследования и отладки динамически реконфигурируемых систем /

М.Ю. Ивленков, А.К. Филиппов // Электроника, информатика и управление: сб. науч. тр. преподавателей, сотрудников и аспирантов. – 2004. – № 5. – С. 8 – 13.

21. *Китаев, А*. Классические и квантовые вычисления / А. Китаев, А. Шень, М. Вялый. – М.: МЦНМО, ЧеРо, 1999. – 192 с. – ISBN 5-900916-35-9.

22. *Ловчиков, А.Н.* Коррекция эксцентрисета орбиты космического аппарата с магнитодинамической тросовой системой / А.Н. Ловчиков, А.П. Ефимов // Изв. вузов. Сер. Приборостроение. – 2004. – № 4. – С. 20 – 23.

23. *Майоров, С.А.* Структура электронных вычислительных машин / С.А. Майоров, Г.И. Новиков. – Л.: Машиностроение, 1979. – 384 с.

24. *Месарович, Д.* Общая теория систем: математические основы / Д. Месарович, Я. Такахара. – М.: Мир, 1978. – 312 с.

25. Методы робастного, нейро-нечеткого и адаптивного управления: учебник / под ред. Н.Д. Егупова. – 2-е изд. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. – 744 с. – ISBN 5-7038-2030-8.

26. *Моисеев, Н.Н.* Математические задачи системного анализа / Н.Н. Моисеев. – М.: Наука, 1981. – 488 с.

27. *Осипов, Л.А*. Обработка сигналов на цифровых процессорах. Линейно-аппроксимирующий метод / Л.А. Осипов. – М.: Горячая линия-Телеком, 2001. – 112 с.

28. *Прангишвили, И.В.* Нанотехника, наноиндустрия, микросистемы / И.В. Прангишвили, А.Г. Алексенко, Р.Р. Бабаян // Датчики и системы. – 2002. – № 7. – С. 57 – 65.

29. *Рабинович, З.Л.* Типовые операции в вычислительных машинах / З.Л. Рабинович, В.А. Раманаускас. – Киев: Техника, 1980. – 264 с.

30. *Руфицкий, М.В.* Оценка эффективности применения динамически реконфигурируемого сопроцессора / М.В. Руфицкий, М.Ю. Федотов // Электроника, информатика и управление: сб. науч. тр. преподавателей, сотрудников и аспирантов. – 2001. – № 2. – С. 59 – 64.

31. *Руфицкий, М.В.* Применение ПЛИС в качестве динамически реконфигурируемого сопроцессора / М.В. Руфицкий, М.Ю. Федотов // Электроника, информатика и управление: сб. науч. тр. преподавателей, сотрудников и аспирантов. – 2000. – № 1 – С. 158 – 163.

32. *Руфицкий, М.В.* Адаптивное устройство вычисления прямых тригонометрических функций / М.В. Руфицкий, А.К. Филиппов // Актуальные проблемы радиоэлектроники и телекоммуникаций : материалы всерос. науч.-техн. конф. – Самара: Изд-во СГАУ, 2004. – С. 20 – 22.

33. *Они же.* Базовые принципы построения адаптивных вычислительных устройств / М.В. Руфицкий, А.К. Филиппов // Проектирование и технология электронных средств. – 2003. – № 2. – С. 2 – 5.

34. Сертификат № 846 от 18 мая 2004 года. Операционная система реального времени QNX версии 4.25 (изделие КПДА.00001-01) / http://www.fstec.ru/_doc/reestr_sszi/_reestr_sszi.xls. Федеральная служба по техническому и экспортному контролю Российской Федерации. Система сертификации средств защиты информации по требованиям безопасности информации № РОСС RU.0001.01БИ00. Государственный реестр сертифицированных средств защиты информации.

35. Сертификат № 906 от 27 октября 2004 года. Защищенная операционная система реального времени «QNX» (изделие КПДА.00002-01) / http://www.fstec.ru/_doc/reestr_sszi/_reestr_sszi.xls. Федеральная служба по техническому и экспортному контролю Российской Федерации. Система сертификации средств защиты информации по требованиям безопасности информации № РОСС RU.0001.01БИ00. Государственный реестр сертифицированных средств защиты информации.

36. *Слик, А.* Устройство адаптивного сжатия данных / А. Слик, А.К. Филиппов // Проектирование и технология электронных средств. – 2003. – № 4. – С. 7 – 12.

37. *Солонина, А.И.* Алгоритмы и процессоры цифровой обработки сигналов / А.И. Солонина, Д.А. Улахович, Л.А. Яковлев. – СПб.: БХВ-Петербург, 2001. – 464 с. – ISBN 5-94157-065-1.

38. Специализированные ЦВМ: учеб. для вузов / В.Б. Смолов [и др.]. – М.: Высш. шк., 1981. – 279 с.

39. *Стешенко, В.Б.* ПЛИС фирмы «Altera»: элементная база, система проектирования и языки описания аппаратуры / В.Б. Стешенко. – М.: Додэка-XXI, 2002. – 576 с. – ISBN 5-94020-001-X.

40. *Угрюмов, Е.П.* Цифровая схемотехника / Е.П. Угрюмов. – СПб.: БХВ-Санкт-Петербург, 2000. – 528 с. – ISBN 5-8206-0100-9.

41. *Уидроу, Б.* Адаптивная обработка сигналов / Б. Уидроу, С. Стирнз; пер. с англ. – М.: Радио и связь, 1989. – 440 с.

42. *Федотов, М.Ю.* Сравнительный анализ режимов динамического реконфигурирования ПЛИС / М.Ю. Федотов // Электроника, информатика и управление: сб. науч. тр. преподавателей, сотрудников и аспирантов. – 2001. – № 2. – С. 64 – 69.

43. *Филиппов, А.К.* Базовые принципы построения комплементарных вычислительных устройств / А.К. Филиппов // Электроника, информатика и управление: сб. науч. тр. преподавателей, сотрудников и аспирантов. – 2004. – № 5. – С. 4 – 8.

44. *Он же*. Высоконадежные динамически реконфигурируемые системы обработки информации для ответственных применений / А.К. Филиппов // Проектирование и технология электронных средств. – 2008. – № 2. – С. 2 – 9.

45. *Он же.* Динамически реконфигурируемые системы как средство реализации алгоритмов адаптивной обработки цифровых сигналов / А.К. Филиппов // Труды Владимирского государственного университета. Вып. 1. Информационно-телекоммуникационные технологии и электроника. – Владимир: Изд-во Владим. гос. ун-та, 2006. – С. 34 – 39.

46. *Он же.* Исследование возможности применения способа непосредственной проверки сходимости для расчета значений прямых тригонометрических функций / А.К. Филиппов // Проектирование и технология электронных средств. – 2004. – № 2. – С. 50 – 55.

47. *Он же.* К вопросу о математическом моделировании динамически реконфигурируемых систем обработки информации / А.К. Филиппов // Проектирование и технология электронных средств. – 2008. – № 4. – С. 35 – 44.

48. *Филиппов, А.К* Обработка информации на основе системы адаптивного вычисления прямых тригонометрических функций / А.К. Филиппов // Проектирование и технология электронных средств. – 2005. – № 1. – С. 2 – 8.

49. *Он же.* Основы математического моделирования динамически реконфигурируемых систем обработки информации / А.К. Филиппов // Проектирование и технология электронных средств. – 2008. – № 1. – С. 38 – 44.

50. *Он же.* Перспективы развития динамически реконфигурируемых систем / А.К. Филиппов // Проектирование и технология электронных средств. – 2005. – № 4. – С. 27 – 31.

51. *Он же.* Современные архитектуры динамически реконфигурируемых систем обработки информации / А.К. Филиппов // Проектирование и технология электронных средств. – 2007. – № 2. – С. 2 – 9.

52. *Филиппов, А.К.* Проектирование цифровых устройств на основе динамически реконфигурируемых систем обработки информации: метод. указания к лаб. работам по дисциплине «Аналоговая и цифровая электроника» / А.К. Филиппов, М.Ю. Ивленков. – Владимир: Изд-во Владим. гос. ун-та, 2008. – 48 с.

53. *Филиппов, А.К.* Проектирование систем адаптивного сжатия данных: формализация задач и анализ вариантов реализации / А.К. Филиппов, В.А. Руфицкий // Проектирование и технология электронных средств. – 2006. – № 3. – С. 14 – 19.

54. *Фрадков, А.Л.* Основы математического моделирования: системный анализ и построение моделей: учеб. пособие / А.Л. Фрадков. – Л.: Ленингр. механ. ин-т, 1989. – 88 с.

55. *Чернов, В.Ю.* Надежность авиационных приборов и измерительно-вычислительных комплексов: учеб. пособие / В.Ю. Чернов, В.Г. Никитин, Ю.П. Иванов. – СПб: СПбГУАП, 2004. – 96 с. – ISBN 5-8088-0100-1.

56. *Arnold, J.* Splash II / J. Arnold, D. Buell, E. Davis // Proceedings of the 4th ACM Symposium of Parallel Algorithms and Architectures. – 1992. – P. 316 – 322.

57. *Fey, D.* Specification for a reconfigurable optoelectronic VLSI processor suitable for digital signal processing / D. Fey, B. Kasche, C. Burkert, O. Tschäche // Applied Optics. – 1998. – №. 2. – V. 37. – P. 284 – 295.

58. *Filippov, A.K.* Adaptive Computing Systems: Definition and Mathematical Background / A.K. Filippov // Proceedings of the 8th International Conference "Perspective technology in the mass media – PTMM'2009". – 2009. – P. 86 – 89.

59. *Filippov, A.K.* Application-specific hardware for the implementation of texture analysis algorithms / A.K. Filippov, V.A. Rufitskiy, M.Yu. Ivlenkov [et al] // Proceedings of the 7th International Conference "Perspective technology in the mass media – PTMM'2007". – 2007. – P. 105 – 108.

60. *Filippov, A.K.* DSP- and FPGA-based Embedded Solutions for Implementing Image Processing Algorithms / A.K. Filippov, V.A. Rufitskiy, V.I. Chukhno [et al] // Proceedings of the 10th International Conference "Digital Signal Processing and its Applications". – 2008. – P. 592 – 596.

61. *Filippov, A.K.* On the Impact of the Computational Properties of Image Processing Algorithms on their Implementation / A.K. Filippov, V.A. Rufitskiy, V.I. Chukhno [et al] // Proceedings of the 9th International Conference "Digital Signal Processing and its Applications". – 2007. – P. 478 – 482.

62. *Filippov A.K.* An Adaptive Wavelet-based Video Compression Algorithm for Very Low Bitrate Applications in Digital Broadcasting Systems / A.K. Filippov, V.A. Rufitskiy, M.V. Rufitskiy [et al] // Proceedings of the 7th Workshop "Digital Broadcasting". – 2006. – P.73 – 85.

63. *Flynn, M.* Some Computer Organizations and Their Effectiveness / M. Flynn // IEEE Transactions on Computers. –1972. – № C-21. – P. 948 – 960.

64. *Fross, B.* WILDFIRE(tm) Heterogeneous Adaptive Parallel Processing Systems / B. Fross, D. Hawver, J. Peterson // The 12th International Parallel Processing Symposium. – 1998. – P. 6 – 11.

65. *Hauck, S.* Configuration prefetch for single-context reconfigurable coprocessors / S. Hauck // Proceedings of ACM/SIGDA International Symposium on Field Programmable Gate Arrays. – 1998. – P. 65 – 74.

66. *Kim, B.-J.* Low Bit-Rate Scalable Video Coding with 3D Set Partitioning in Hierarchical Trees (3D SPIHT) / B.-J. Kim, Z. Xiong,

W.A. Pearlman // IEEE Transactions on Circuits and Systems for Video Technology. – 2000. – № 6. – P. 1374 – 1387.

67. *Lan-Da Van*. Systolic Architecture Design //www.cs.nctu.edu.tw/~ldvan/teaching/vlsidsp/VLSIDSP_CHAP7.pdf.

68. *Mencer, O.* Hardware Software Tri-Design of Encryption for Mobile Communication Units / O. Mencer, M. Morf, M.J. Flynn // IEEE International Conference on Acoustics, Speech and Signal Processing. – 1998. – № 5. – P. 3045 – 3048.

69. *Niemann, H.* Methoden der Mustererkennung / H. Niemann. – Frankfurt: Akademische Verlagsgesellschaft, 2003. – 468 S.

70. *Said, A.* A new, fast and efficient image Codec Based on Set Partitioning in Hierarchical Trees / A. Said, W.A. Pearlman // IEEE Transactions on Circuits and Systems for Video Technology. – 1996. – № 6. – P. 243 – 250.

71. *Vuillemin, J.* Programmable Active Memory: Reconfigurable Systems Come of Age / J. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. Touati, P. Boucard // IEEE Transactions on Very Large Scale Integration Systems. – 1996. – № 4. – P. 56 – 69.

72. *Wirthlin, M.* A Dynamic Instruction Set Computer / M. Wirthlin, B. Hutchings // Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines. – 1995. – P. 99 – 107.

73. *Wolf, W.* FPGA-Based System Design / W. Wolf. – New Jersey: Prentice Hall PTR, 2004. – P. 576.

74. *Zeidman, B.* Designing with FPGAs and CPLDs / B. Zeidman. – Berkeley: Elsevier, 2002. – P. 220.

ОГЛАВЛЕНИЕ

Учебное издание

ФИЛИППОВ Алексей Константинович

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОЕКТИРОВАНИЯ ДИНАМИЧЕСКИ РЕКОНФИГУРИРУЕМЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ

Учебное пособие

Подписано в печать 25.02.10.
Формат 60x84/16. Усл. печ. л. 6,97. Тираж 100 экз.
Заказ .
Издательство
Владимирского государственного университета.
600000, Владимир, ул. Горького, 87.