ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

7(155) 2009

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с ноября 1995 г.

УЧРЕДИТЕЛЬ
Издательство "Новые технологии"

СОДЕРЖАНИЕ

Информация о журнале доступна по сети Internet по адресу http://www.**informika.ru/text/magaz/it/** или **http://novtex.ru/IT.**

Журнал включен в систему Российского индекса научного цитирования.

Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

УДК 004.3

А. А. Сигарев, научный сотрудник,
Институт проблем моделирования в энергетике
им. Г. Е. Пухова НАН Украины, Киев,
e-mail: em@ipme.kiev.ua,
london@diawest.net.ua

Методология упразднения межпроцессорного обмена в МВС со множественным потоком команд

Показана принципиальная возможность упразднения межпроцессорного обмена в многопроцессорных вычислительных системах (МВС) со множественным потоком команд. Единственной аппаратной базой полного его исключения может служить только специальным образом организованная двухпортовая память, названная памятью передачи данных. Функции обмена в предлагаемом подходе может выполнять любая адресная арифметико-логическая операция при условии попадания ее адресной части в определенный зафиксированный диапазон адресного пространства. Упразднение межпроцессорного обмена ведет к увеличению производительности МВС, а интенсивность межпроцессорного обмена перестает играть роль критического фактора при распараллеливании программ.

Ключевые слова: *межпроцессорный обмен, ортогональные многопроцессорные системы, конвейерные системы, синхронизация.*

Введение

Если кратко охарактеризовать весь полувековой период развития МВС, то необходимо отметить, что проблема межпроцессорного обмена (МО) всегда была актуальной. Были периоды кризиса, когда неудовлетворительное состояние проблемы МО становилось тормозом в развитии систем, но даже в свои наилучшие времена проблема МО редко получала удовлетворительную оценку специалистов. В 1996 г. прошлого века автор широко известной классификации МВС Флинн открыто признавал состояние МО неудовлетворительным [1] и даже призывал к осуществлению "мозгового штурма" данной проблемы.

Какой оценки заслуживает сегодняшнее состояние проблемы МО? Современные авторы отмечают чрезмерную громоздкость дисциплины обменов, которая во многих случаях не обеспечивает управляемость вычислительного процесса. Выполненные исследования и многочисленные тестирования [2] показали, что на реальных задачах в многопроцессорных системах с массовым параллелизмом производительность составляет 5—40 % от пиковой. На некоторых очень важных классах алгоритмов производительность достигает 0,1—1 % от пиковой. Основные причины такого резкого падения реальной производительности обусловлены следующим:

- задержкой выполнения операций с оперативной памятью;
- задержкой выполнения операций с коммуникационной средой;
- задержкой вследствие рассинхронизации процессов и потерь на операционную систему;
- задержкой выполнения команд, обусловленные их зависимостью по управлению и по данным.

Таким образом, большинство негативных факторов, влияющих на производительность систем, прямо или косвенно имеют отношение к МО.

В целом такое положение нельзя считать удовлетворительным, а проблема МО продолжает оставаться актуальной. Тем не менее, вопрос об упразднении МО никогда не стоял и не ставится в наше время. Объективности ради надо отметить, что механизмов такого упразднения до сих пор предложено не было.

Понятие "упразднение МО"

Правомерна ли вообще постановка вопроса об упразднении МО? После распараллеливания исходной программы в нее вводятся функции обмена, которые включают целую систему громоздких взаимодействий каждого параллельного фрагмента программы с сетью, операционной системой и приемником/отправителем информации. Исполнение всей совокупности процедур и протоколов, задействованных при МО, приводит к существенным потерям системной производительности. Упразднение МО устраняет эти потери.

Наша цель — показать, что набор инструкций исходной программы достаточен для ее реализации в распараллеленном виде, что равнозначно полному упразднению МО. В качестве нашей рабочей модели выберем гипотетическую вычислительную систему (ВС), в состав которой входит два вычислительных узла ВУ1 и ВУ2 (рис. 1) и среда передачи данных. Каждый ВУ является ти-

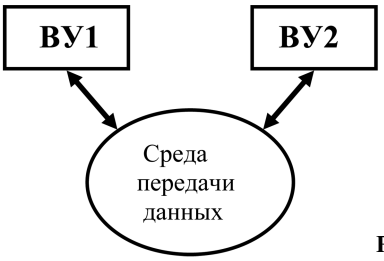


Рис. 1. Гипотетическая ВС

повой машиной фон-неймановского типа, имеющей в своем составе процессор, ОЗУ и устройства ввода-вывода. Кроме того, полагаем, что наименьшим адресуемым элементом памяти является машинное слово. В системе нет единого адресного пространства, а каждый из ВУ имеет доступ только в свое собственное ОЗУ.

Сформируем среду передачи данных таким образом, чтобы полностью исключить МО. Такое исключение предполагает создание некоторого нового механизма, основным свойством которого будет полное отсутствие:

- программных процедур, реализующих передачу данных и ее синхронизацию;
- временных затрат на их осуществление.

Исключение функций обмена

Для иллюстрации предлагаемой методологии будем использовать загрузочный модуль некоторой программы Ex (рис. 2, *а*), которая структурно может быть разбита на два последовательных фрагмента: фрагмент 1 и фрагмент 2. Полагаем для простоты, что в программе нет обращений ни к подпрограммам, ни к функциям, как и нет передач управления из одного фрагмента в другой. Указанные фрагменты допускают и одновременное исполнение. Данные при приведенном разбиении программы могут быть упорядочены и расчленены на три части. Данные 1 предназначены только для фрагмента 1 и данные 2 — соответственно для фрагмента 2. Общие данные используются в обоих фрагментах.

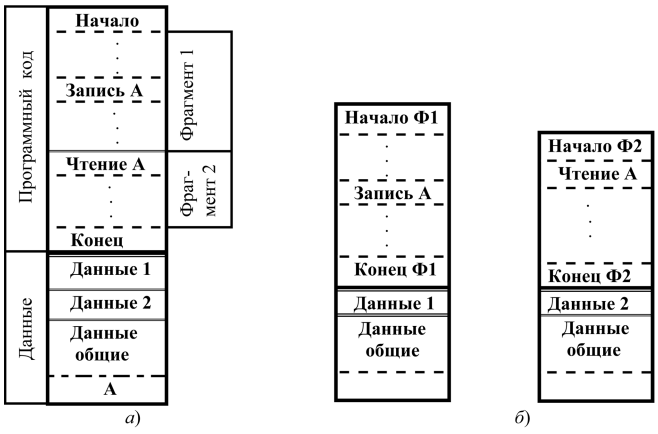


Рис. 2. Структура загрузочного модуля исходной программы (*а*), структура загрузочных модулей после распараллеливания исходной программы (*б*)

Фрагмент 1 в некоторой точке своего исполнения осуществляет запись операнда А в последнюю ячейку сегмента общих данных, который затем считывается фрагментом 2 программы. В данном случае фрагменты взаимодействуют через одну и ту же ячейку памяти. Это естественный и непосредственный способ обмена между взаимосвязанными программными фрагментами, используемый в последовательных программах.

Распараллелим данную программу, как показано на рис. 2, *б*. При этом и фрагмент 1 (Ф1) и фрагмент 2 (Ф2) программы извлекаются из последовательной программы в первозданном виде, т. е. в них остаются неизменными как набор инструкций, так и последовательность их исполнения. Изменению могут подлежать только адресные части инструкций. Каждый из фрагментов снабжается необходимым для его автономного исполнения набором данных, и в итоге получаем два загрузочных модуля для упомянутых ветвей программы. Пока не будем затрагивать местоположения операнда А, однако ни в одном из сегментов общих данных место для него не резервируется.

Представим, что программа Ex после распараллеливания загружается в систему, приведенную выше. Для определенности пусть Ф1 и Ф2 загружаются в ОЗУ соответственно ВУ1 и ВУ2. Для корректного исполнения рассматриваемой программы необходимо операнд А из ВУ1 переслать в ВУ2 таким образом, чтобы при этом не появилось ни одной новой инструкции ни в одном из указанных программных фрагментов. При этом исключается какое-либо вмешательство ОС и протоколов обмена любых видов. Последняя ячейка сегмента "Данные общие" в данном случае для наших целей бесполезна.

Как уже было отмечено и показано, фрагменты в нераспараллеленной программе взаимодействуют через память. Таким образом, необходимыми условиями отсутствия процедуры МО являются:

- осуществление межпроцессорного обмена через память;
- доступность данной памяти для всего набора адресных арифметико-логических инструкций обоих процессоров;
- обязательное наличие двух портов у обозначенной выше памяти.

При всей тривиальности сделанных выводов из них следует, что большой и совершенно неиспользуемый потенциал совершенствования МО заключен именно в двухпортовой памяти.

Первым непосредственным следствием полученных выводов является частичная конкретизация обозначенной выше среды обмена, с учетом

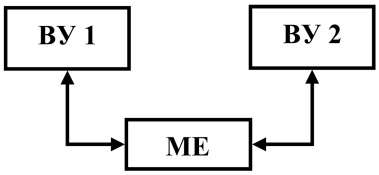


Рис. 3. Схема передачи данных через память МЕ

которой система принимает вид, приведенный на рис. 3, где МЕ — двухпортовая память.

Память МЕ должна являться частью ОЗУ каждого из вычислительных узлов, и под нее в адресном пространстве каждого из процессоров должны быть отведены равные и фиксированные диапазоны адресов. Таким образом, для обращения к МЕ необходимо сформировать стандартный набор сигналов, адрес и (в случае записи) операнд и подать на соответствующие входы МЕ. При такой организации системы передача операнда А из ВУ1 в ВУ2 вполне очевидна: фрагмент 1 записывает его в МЕ, а ВУ2 естественным образом считывает его. МО можно считать полностью упраздненным только в том случае, если и запись, и чтение операнда будут осуществляться теми же самыми командами, которыми они осуществляются в исходной программе. Исполнительный адрес при записи и чтении операнда соответствующими процессорами должен быть одинаковым и строго фиксированным и не зависеть от расположения программных фрагментов в ОЗУ своих вычислительных узлов.

Единственное условие, которое необходимо при этом соблюсти, состоит в том, чтобы чтение операнда А начиналось только после его записи, т. е. необходимо ввести в саму память МЕ аппаратный механизм синхронизации передачи операнда, который вообще не расходует процессорное время. В соблюдении этого условия и заключается существо всей этой проблемы.

Процесс обращения к памяти МЕ посредством адресных арифметико-логических операций будем называть *локальной передачей данных* (ЛПД).

Принципы синхронизации

При одновременной активизации обоих фрагментов программы в рассматриваемой системе Ф2 приступит к чтению операнда А прежде, чем произойдет его реальная запись в память. Именно такая ситуация изображена на рис. 2, *б*. Во избежание сбоя вычислительного процесса необходимо задержать исполнение операции чтения операнда, но любой известный механизм для такой задержки будет содержать программную процедуру.

Механизм задержки исполнения адресных операций при ЛПД в МВС, не имеющий программной процедуры и не требующий затрат полезного компьютерного времени, будем называть *механизмом синхронизации передачи/приема* (МСП).

Такой механизм необходимо еще создать. Начнем с алгоритма его функционирования. Процессор, начиная обращение к МЕ адресной инструкцией, не знает, допустима ли эта операция в принципе в данный момент времени. Для предотвращения сбоя вычислений уже в процессе исполнения начатой операции необходимо как раз и установить эту допустимость. Если операция допустима, то исполнение ее происходит в штатном режиме. Если она недопустима, необходимо:

- заблокировать обращение к модулю памяти с данного порта;
- заблокировать окончание операции во избежание модификации внутренних регистров и счетчика команд процессора;
- осуществить штатное завершение начатой операции после разблокирования данного порта.

Цикл исполнения каждой инструкции в процессоре содержит несколько тактов. Такты, в которые происходят чтение команды и дешифрация кода операции и адреса, могут быть исполнены, и никакой угрозы вычислительному процессу это не представляет. Исполнение такта, в котором может произойти недопустимое в данный момент времени чтение/запись, приведет к сбою всего вычислительного процесса. Единственно возможный способ приостановить такое катастрофическое развитие ситуации — затянуть во времени данный тактовый импульс, заставить "зависнуть" процессор на такте чтения/записи до тех пор, пока не будет выполнена запись/чтение со стороны другого порта модуля памяти, в результате чего разблокируется данная ячейка. Только после этого должно произойти штатное завершение начатой операции.

Для реализации МСП каждая ячейка двухпортовой памяти делается переключаемой и может в любой произвольный момент времени быть подключена только к одному из портов памяти, кроме того, в нее вводится аппаратный тег. Значение тега указывает номер порта памяти МЕ, к которому подключена в данный момент времени адресуемая ячейка памяти. Операции записи в обязательном порядке меняют значение тега, осуществляя тем самым автоматическое переключение ячейки с записанным в нее операндом на другой порт памяти МЕ.

Переключение ячейки может быть выполнено только со стороны порта, к которому в данный момент времени подключена ячейка. Это означает, что у процессора нет механизма захвата ячейки памяти, но есть механизм передачи ее другому устройству. Это принципиально важный момент, ибо наличие механизма захвата не позволяет устранить МО.

При каждом обращении к ячейке аппаратно осуществляется анализ ее тега, который запускается адресной частью команды и протекает параллельно с самим процессом обращения. В зависимости от значения тега происходит одно из двух:

- запись/чтение и штатное завершение операции;
- осуществление режима ожидания в процессе исполнения начатой операции с данной ячейкой.

Если ячейка в текущий момент времени подключена к данному порту, то имеет место типовое обращение к памяти. В противном случае формируется сигнал L, который, поступая в соответствующий процессор, затягивает во времени текущий такт операции. Это состояние будем именовать *режимом ожидания в процессе исполнения начатой операции чтения/записи* (РО). РО длится до момента подключения адресуемой ячейки памяти к данному порту. После завершения начатой операции запись/чтение снимается сигнал L и восстанавливается штатный режим работы процессора. Все приведенные процедуры остаются невидимыми для программиста и не имеют отображения в программе.

Двухпортовую память, имеющую механизм МСП, назовем памятью передачи данных (МЕ).

Таким образом, модуль памяти передачи данных МЕ (рис. 4) имеет специальную организацию и предназначен для информационного обмена только двух процессоров (устройств). Он обладает функциональным свойством вырабатывать два управляющих сигнала L1 и L2, по одному на каждый порт памяти, которые призваны решать проблему блокировки памяти и синхронизации ЛПД, полностью освобождая от этих функций устройство управления процессора, операционную систему и программиста. Для этих целей в каждую ячейку памяти введен аппаратный тег, значение которого указывает номер порта памяти, со стороны которого разрешена операция с данной ячейкой памяти. Операции записи заканчиваются обязательным изменением значения тега ячейки памяти. Процессор, изменивший значение тега, переключает тем самым данную ячейку памяти на другой порт МЕ, передавая, таким образом, записанное в нее слово сопряженному процессору.

Далее при использовании аббревиатуры МЕ будет всегда подразумеваться память, интерфейс которой приведен на рис. 4.

Могут быть две модификации памяти передачи данных:

- более универсальная организация МЕ, когда значения тега со стороны обоих портов изменяют операции записи. В этом случае имеет место не что иное, как дуплексный тип ЛПД;

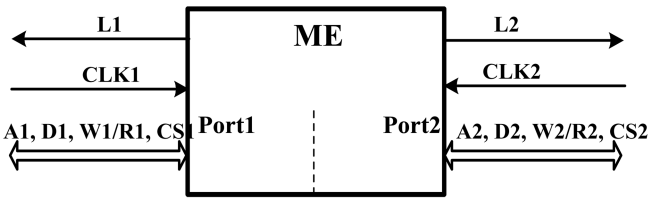


Рис. 4. Схема интерфейса модуля памяти передачи данных

- в случае, если значения тега на одном из портов МЕ изменяют операции записи и операции чтения на другом, то будет иметь место симплексный тип ЛПД.

Исходное значение МЕ достигается начальной установкой. Рассмотрим, как будет осуществляться передача операнда А от Ф1 через память МЕ в Ф2 (см. рис. 2, *б*). Допустим, что после начальной установки ко всем ячейкам МЕ разрешен доступ только со стороны 1-го порта, а любая попытка обращения со стороны 2-го порта переводит соответствующий процессор в РО. В случае одновременной активизации в системе параллельных фрагментов ВУ2 перейдет в РО при попытке чтения еще не записанного в МЕ операнда А на такте обращения к самой ячейке памяти. В этом состоянии процессор будет находиться до тех пор, пока ВУ1 не запишет в выбранную ячейку операнд А. Операция записи помимо этого изменит значение тега и передаст ячейку вместе с записанным в нее операндом А узлу ВУ2. С этого момента сразу возобновляется начатое чтение уже реального операнда процессором ВУ2 до штатного завершения этой операции. Если бы операнд А был записан своевременно в МЕ, то никаких видимых признаков "обмена" в рассматриваемом примере обнаружить не представлялось бы возможным. Таким образом, любая адресная команда может одновременно являться и командой ЛПД другому процессору и исключать в этом случае дополнительные затраты времени на МО, которые имеют место в любых известных на сегодня методах МО.

Из сказанного выше следует, что традиционный набор адресных арифметико-логических операций является эффективным средством ЛПД между процессорами системы, имеющими непосредственную связь. Самое главное состоит в том, что даже максимально возможная интенсивность передачи вообще не требует никаких затрат времени на свое осуществление. Предложенный механизм проще и эффективнее существующих технологий межпроцессорного обмена. При реализации предложенного механизма нет неразрешимых проблем принципиального характера. Не вызывает сомнения, что замена существующих сетевых технологий обмена предложенной памятью там, где это представляется возможным, может существенно повысить производительность МВС.

В порядке информирования необходимо отметить, что модель такой памяти изготовлена и проверена в действии.

Применение памяти передачи данных

Память передачи данных МЕ и заложенные в нее идеи могут найти весьма широкое применение.

Создание конвейерных систем. Предложенная память МЕ незаменима при организации макроконвейерных систем [3] и вообще любых типов конвейеров — синхронных и асинхронных. В данном случае организация асинхронных конвейеров формально ничем не будет отличаться от синхронных, так как рассинхронизация ступеней конвейера в принципе становится невозможной.

Конвейеризация широко используется при разработке процессорных кристаллов. Не исключено, что использование идей, заложенных в памяти МЕ, может существенно упростить структурную организацию и уменьшить аппаратную сложность внутри кристальных конвейеров.

Наконец, упрощается организация систем специализированных, обладающих одновременно наличием временного и пространственного параллелизма, в которых особенно остро может стоять проблема синхронизации обмена.

Организация систем класса МКМД. Для создания N-процессорной системы класса МКМД необходимо $N(N-1)/2$ модулей МЕ объединить в треугольную матрицу размерности $N \times N$ так, чтобы к каждому отдельному модулю МЕ доступ имела только одна единственная пара процессоров (рис. 5). Приведенная организация идеально подходит для решения численными методами систем уравнений различных видов [4—9]. Она относится к классу ортогональных многопроцессорных систем (ОМС). При этом даже максимально сильная связь по данным не будет сказываться на производительности системы, более того возможен ее рост выше линейного за счет увеличения общего числа сверхоперативных регистров и кэшей в целом на систему.

В случае организации по такой схеме четырехядерного процессора на обмен не будет расходоваться ни одного такта полезного системного времени. Такая организация, кроме того, позволяет процессору одной командой записи занести один и тот же операнд по некоторому адресу сразу трех модулей памяти МЕ, т. е. за один цикл записи все процессоры смогут обменяться операндами.

Возможности предлагаемого метода требуют пересмотра и принципов структурной организации систем, и принципов распараллеливания программ. Однородная четырехвалентная сеть на двумерной решетке (рис. 6, *а*) является одной из весьма распространенных структурных организаций систем с

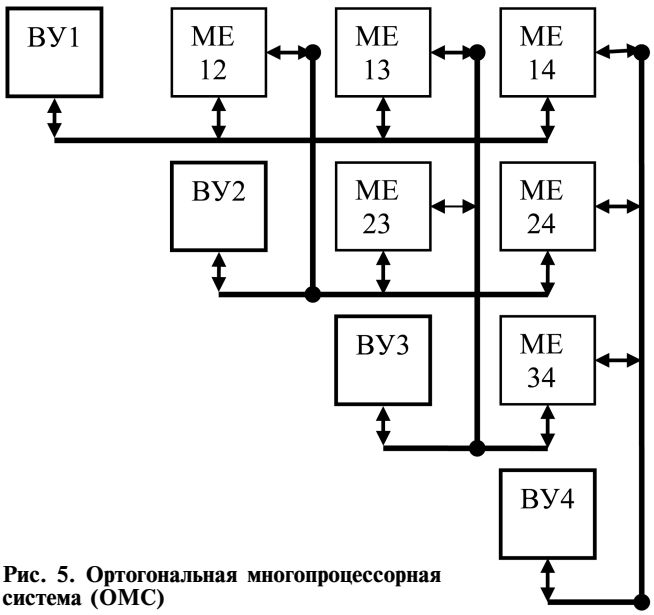
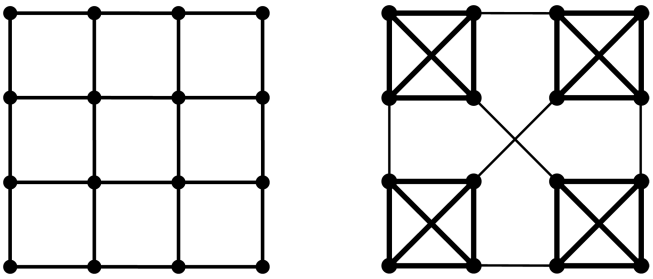


Рис. 5. Ортогональная многопроцессорная система (ОМС)



а) *б*)

Рис. 6. Однородная четырехвалентная сеть на двумерной решетке (*а*), однородная иерархическая сеть (*б*)

массовым параллелизмом. Однако наибольший эффект от применения ЛПД следует ожидать от более универсальной четырехвалентной однородной иерархической сети, фрагмент которой с числом узлов $n = 8$ приведен на рис. 6, *б*. Ее преимущество состоит в том, что она состоит из четырех групп вершин, имеющих связь " каждая с каждым", т. е. между этими узлами можно осуществлять ЛПД, а не МО, и замыкать на эти узлы самые интенсивные информационные потоки. Следовательно, распараллеливание задач необходимо начинать с поиска параллельных частей задачи, имеющих сильную связь по данным, а не наоборот, как это делается сейчас. Связь между группами узлов можно осуществлять по сетевым технологиям. Число уровней иерархии очень просто наращивается при неизменной валентности узлов в отличие от гиперкубов, в которых наращивание влечет за собой рост валентности каждого узла. Рассматриваемая сеть отличается очень простой и удобной адресацией и может иметь любую валентность $p > 4$.

Организация памяти FIFO. Память FIFO, организованная на базе памяти ME, обладает уникальными свойствами. Ее можно рассматривать, как имеющую бесконечную емкость. С точки зрения формальной логики такая FIFO, с одной стороны, всегда готова для записи и, с другой стороны, в ней всегда имеется записанный операнд, так как для записи и чтения не требуется никакой формальной проверки состояния такой памяти.

Чтение операнда из памяти может начинаться, когда этот операнд еще не записан. Аналогично запись операнда может начинаться в тот момент, когда физически память переполнена.

Список литературы

1. **Flynn M. J.** Parallel processors were the future and may yet be // Computer. 1996. Vol. 29. № 12. P. 12—24.
2. **Митрофанов В., Слуцкий А., Ларионов К., Эйсымонт Л.** Направление развития отечественных высокопроизводительных систем // Открытые системы. 2003. № 5.
3. **Сигарев А. А., Душеба В. В.** Способ упразднения межпроцессорного обмена в макроконвейерах // Электронное моделирование. 2006. Т. 28, № 6. С. 71—89.
4. **Хуан К.** Перспективные методы параллельной обработки и архитектуры суперЭВМ // ТИИЭР. 1987. Т. 75, № 10. С. 4—41.
5. **Hwang K., Tseng P. S., Kim D.** An Ortogonal Multiprocessor for Parallel Scientific Computations // IEEE Transactions on Computers. January. 1989. Vol. 38. № 1. P. 120—124.
6. **А. с. № 1608700 СССР,** МКИ G 06 F 15/80 // Матричная вычислительная система / В. Ф. Евдокимов, И. Ф. Зубенко, А. Т. Манохин, А. А. Сигарев / № 4663952. Заявлено 20.03.1989 г.
7. **А. с. № 1839261** СССР, МКИ G 06 F 15/16 // Ортогональная многопроцессорная система / А. А. Сигарев / № 4821405. Заявлено 07.05.1990 г.
8. **Сигарев А. А., Душеба В. В.** Методы организации ортогональных многопроцессорных систем. I // Электронное моделирование. 1999. Т. 21. № 6. С. 57—65.
9. **Сигарев А. А., Душеба В. В.** Методы организации ортогональных многопроцессорных систем. II // Электронное моделирование. 2000. Т. 22. № 1. С. 59—73.

УДК 519.688; 004.451.31

Д. Л. Аверичева, науч. сотр.,
А. С. Семенов, нач. сектора,
e-mail: semenov@nicevt.ru
А. С. Фролов, нач. сектора,
ОАО "НИЦЭВТ"

Поиск вширь в графе на суперкомпьютере с мультитредово-потоковой архитектурой

Предложены новые алгоритмы эффективного решения на российском мультитредово-потоковом (МТП) суперкомпьютере задачи поиска вширь в графе, которая характеризуется интенсивной нерегулярной работой с памятью. Результаты, полученные при выполнении разработанных алгоритмов на программной имитационной модели МТП-суперкомпьютера, сравниваются с наилучшими известными результатами выполнения поиска вширь в графе на разных суперкомпьютерах.

Ключевые слова: *поиск вширь в графе, обработка графов, суперкомпьютинг, распределенная общая память, мультитредовость.*

Введение

Важность задач обработки больших графов возрастает на протяжении последних лет. Для оценки возможности использования вычислительных систем для решения таких задач разрабатываются новые тестовые программы и методики. Например, в рамках программы DARPA HPCS разработан пакет SSCA2 [1, 2].

Эффективное решение графовых задач требует как новых программных решений, так и аппаратных архитектур [3, 4]. Это связано с тем, что хранение больших графов невозможно в памяти одного вычислительного узла, кроме того, графовые задачи характеризуются интенсивной нерегулярной работой с памятью, а это приводит к тому, что такие задачи плохо решаются на традиционных суперкомпьютерах кластерного типа.

В ОАО "НИЦЭВТ" при участии авторов статьи ведется проект создания суперкомпьютера с мультитредово-потоковой (МТП) архитектурой и аппаратной поддержкой распределенной общей памяти — МТП-суперкомпьютера [5]. Он предназначен для эффективного решения задач с интенсивной нерегулярной работой с памятью и не уступает традиционным суперкомпьютерам при решении задач другого типа. В данной статье исследуется возможность эффективного решения на МТП-суперкомпьютере одной из графовых задач — задачи поиска вширь в графе.

1. Архитектура МТП-суперкомпьютера

В рамках российского проекта создания МТП-суперкомпьютера разрабатывается оригинальный мультитредово-потоковый микропроцессор и маршрутизатор коммуникационной сети. МТП-суперкомпьютер содержит множество вычислительных узлов, соединенных коммуникационной сетью, с

большой суммарной пропускной способностью при передаче коротких пакетов.

Вычислительный узел (далее просто узел) МТП-суперкомпьютера содержит: сетевой адаптер/маршрутизатор; многоядерный мультитредово-потоковый микропроцессор (варианты J7 и J10) с аппаратной поддержкой трансляции адресов глобально-адресуемой памяти и передачи коротких системных пакетов, реализующих такие обращения; локальную память с большим расслоением на базе стандартных DRAM-модулей.

В процессе исследований рассматривались базовые конфигурации микропроцессоров J7 и J10 [5], параметры используемых в работе конфигураций представлены в табл. 1. Вариант J10 принципиально отличается от варианта J7 аппаратной поддержкой гетерогенной мультитредовости, а также вычислительных моделей типа статических и динамических графов потоков данных, однако эти возможности микропроцессора J10 в данной работе не используются.

Для понимания излагаемого далее материала необходимо пояснить особенности отображения (распределения) адресов сегментов глобально адресуемой виртуальной памяти на локальную физическую память каждого из узлов.

Во-первых, возможно использование циклического отображения с выбором узла по младшим разрядам адреса со скремблированием, при котором каждые следующие восемь слов последовательности подряд идущих виртуальных адресов отображаются на физическую память узла, выбираемого по детерминированной нерегулярной схеме.

Во-вторых, возможно использование отображения виртуальных адресов на физические, когда номер узла задается старшими битами виртуального адреса. Такое отображение называется блочным. В данном случае сегмент распределяется равными порциями (блоками) по узлам, на которые этот сегмент распределен: в физической памяти первого узла располагается первый блок подряд идущих виртуальных адресов, в физической памяти второго узла — второй блок, на последнем узле — последний блок.

Таблица 1

Основные характеристики вариантов многоядерных мультитредово-потоковых микропроцессоров МТП-суперкомпьютера

Параметр	J7-2	J10-4
Частота, ГГц	0,5	2
Число ядер/тредов	2/64	8/128
Количество команд, выдаваемых в ядре на выполнение за такт	4	8
Пиковая производительность процессора, Гфлопс	4	128
Размер кэша, Мбайт/way	1/4	4/16
Пропускная способность кэш-памяти, Гбайт/с	64	256
Пропускная способность DRAM-памяти, Гбайт/с	25,6	204,8
Односторонняя пропускная способность линка сети 4D-тор, Гбайт/с	12	64

Для оценки производительности МТП-суперкомпьютера на тестовых оценочных программах в ОАО "НИЦЭВТ" на языке Charm++ [6] разработана параллельная потактовая имитационная модель, которая хорошо масштабируется по производительности при использовании до 512 узлов суперкомпьютера МВС-100*k*, имеющегося в Межведомственном суперкомпьютерном центре РАН. Отрабатываемые в модели временные диаграммы команд работы с памятью и работы сети максимально приближены к реальным.

2. Поиск вширь в графе

Одним из наиболее популярных алгоритмов обработки графов является поиск вширь (Breadth-First Search, BFS) в графе. Этот алгоритм используется, например, в таких задачах, как поиск всех достижимых вершин из заданной вершины, поиск кратчайшего пути в графе, поиск в графе вершины или ребра с заданными свойствами. По алгоритмам поиска вширь в графе существуют опубликованные данные о производительности на разных архитектурах.

Для заданного графа и заданной вершины r BFS находит все вершины, достижимые через ребра графа от вершины r. Особенностью алгоритма является то, что BFS не должен анализировать вершины, отстоящие от вершины r на расстоянии $s + 1$ ребер до тех пор, пока не проанализирует все вершины, отстоящие от вершины r на расстоянии s ребер. Вершины, отстоящие от исходной вершины r на заданном расстоянии, называются уровнем.

Характеристикой производительности алгоритма BFS по аналогии с [7] будем считать число миллионов пройденных ребер графа в секунду (МЕ/с).

В статье рассматриваются неориентированные графы Пуассона, где вероятность существования ребра между любыми двумя вершинами постоянна. Такие графы часто встречаются в литературе, их просто создавать, распараллеливание алгоритмов решения различных задач над ними наиболее трудно, так как в этих графах отсутствует регулярность. В данной работе графы создаются таким образом, чтобы арности вершин были равномерно-случайными и независимыми числами от 0 до $2k$, где арностью называется число смежных вершин с данной вершиной, k — требуемая средняя арность. Число вершин в графе обозначается N.

Граф задается матрицей смежности, которая во всех вариантах реализации хранится в виде разреженной матрицы в формате CRS с плотным хранением строк ненулевых элементов. Для этого ис-

пользуются два массива: массив VertsIndexes длиной $N + 1$ состоит из адресов первых элементов списков смежности вершин для каждой вершины. Второй массив Edges состоит из списков номеров смежных вершин для каждой вершины. Используется также массив Marked, в котором для каждой вершины хранится признак отметки, была ли эта вершина пройдена в процессе поиска или еще нет.

3. Поиск вширь в графе на одном процессоре

На рис. 1 представлена схема мультитредовой реализации алгоритма *uniform* поиска вширь в графе для одного процессора МТП-суперкомпьютера. Реализация фактически является процедурным описанием поиска вширь с выделением тредовых вычислений и синхронизацией их работы.

Внешний цикл движения по вершинам уровня Q распараллелен по тредам. Каждый тред получает вершину vertex из Q, пробегает список ее соседей E_{vertex}, отмечает те из них, которые еще не были отмечены, и добавляет их на следующий уровень Q_{next}. Проверка отмеченности вершины и, возможно, ее отметка проводится с помощью одной атомарной операции системы команд МТП-суперкомпьютера acswp, которая сравнивает значение элемента Marked[*w*] с нулем и в случае равенства нулю (признак неотмеченности) атомарно записывает единицу. При любом значении Marked[*w*] acswp возвращает его старое значение. Добавление вершины на следующий уровень осуществляется с помощью счетчика вершин следующего уровня $Q_{next\_counter}$, который увеличивается с использованием атомарной операции *afinc*. Использование атомарных операций позволяет синхронизировать доступ тредов к разделяемым ресурсам. Для ожидания завершения обработки вершин текущего уровня происходит барьерная синхронизация тредов, а затем переход на следующий уровень.

```
Q_counter = 1 // инициализация счетчика вершин Q
Q[0] = r // инициализация текущего уровня Q
Marked[r] = 1 // отметка исходной вершины r
while Q_counter >0 // пока уровень не пуст
    Q_next_counter = 0 // обнуление счетчика следующего уровня
    for all vertex ∈ Q do in parallel // параллельная обработка уровня
        for all w ∈ E_vertex do // для всех вершин w, смежных с vertex
            if (!acswp(Marked[w], 0, 1) then Q_next[afinc(Q_next_counter)] = w
        end for
    end for
    Barrier // переход на следующий уровень, обмен Q и Q_next
end while
```

Рис. 1. Схема простейшего алгоритма *uniform* поиска вширь в графе

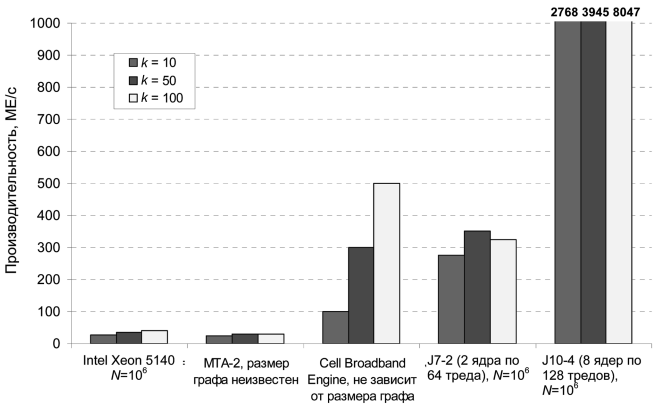


Рис. 2. Производительность на задаче поиска вширь, МЕ/с (миллионов пройденных ребер в секунду); k — средняя арность графа

На рис. 2 дается сравнение производительности алгоритма поиска вширь на разных процессорах [7, 8]. Нерегулярный характер обращений к массивам VertsIndexes, Edges, Marked приводит к промахам в кэш при обработке больших графов, что влечет за собой обращения к DRAM-памяти, пропускная способность которой становится критическим ресурсом. Низкая производительность коммерческого процессора Intel Xeon объясняется слабой подсистемой памяти. Мультитредовость МТП-суперкомпьютера обеспечивает параллелизм обращений и толерантность к задержкам. Благодаря высокому параллелизму (большое число ядер и тредов), поддержанному очень высокой пропускной способностью DRAM-памяти, конфигурация J10-4 показывает рекордно высокую производительность.

Конфигурация J7-2 сравнима по производительности с результатами для процессора Cell, но стоит иметь в виду, что для Cell используется сложный специально разработанный алгоритм [7], а описанная реализация для процессора МТП-суперкомпьютера представляет собой фактически последовательную реализацию с некоторыми изменениями для корректного распараллеливания цикла.

4. Многопроцессорная реализация поиска вширь в графе

Простейший равномерный вариант. Глобально-адресуемая память МТП-суперкомпьютера позволяет запустить практически без изменений мультитредовый алгоритм *uniform* на многих узлах. На каждом уровне вершины из Q равномерно делятся по узлам и, как и в предыдущем случае, обрабатываются всеми тредами параллельно. Массив VertsIndexes при загрузке задачи копируется на каждый работающий узел, массивы Edges,

Marked, Q и Q_{next} хранятся в глобальной памяти и адресуются в режиме скремблирования.

Хотя при небольшом числе узлов (см. ниже рис. 5 и 6) наблюдается ускорение, например, при переходе от 8 узлов к 16 алгоритм неэкономично использует дорогой ресурс — пропускную способность коммуникационной сети — на каждом уровне выполняется в среднем $(2k + 1)|Q| + 2|Q_{next}|$ обращений в глобальную память (обобщенно — глобальных операций), где $|Q|$ — число вершин в Q. Дальнейшие варианты алгоритма по возможности учитывают локализацию и сокращают число обращений в глобальную память.

Распределенный вариант с параллельными фазами. В алгоритме *parallel phases* вершины графа разбиваются по узлам примерно равными порциями по аналогии с работой [9]. Для своих вершин каждый узел имеет локальные массивы Q, Q_{next} и Marked. Схема нового алгоритма дана на рис. 3.

Массив Edges хранится в глобальном блочно-распределенном сегменте. Так как для рассматриваемых в данной статье графов длины списков смежностей вершин имеют равномерное распределение от 0 до $2k$, то можно доказать, что при блочном распределении почти все списки смежности вершин будут лежать в локальной памяти узлов, которым эти вершины принадлежат. Глобально-адресуемая память позволяет прозрачно для программы получить те части списков, которые все-таки попадают в память других узлов. За счет этого достигается простота алгоритма.

```
// инициализация
while  sum_{i=1}^{nodesNumber} Q^i_counter ≠ 0  do in parallel // суммирование по всем узлам
    do phase1
        for all vertex ∈ Q  do in parallel
            for all w ∈ E_vertex  do
                Q^{nodeNumber(w)}_next_tmp [afinc(Q^{nodeNumber(w)}_next_tmp_counter)], synchr = w
            end for
        end for
        Barrier // барьерная синхронизация тредов первой фазы
    end phase1
    do phase2
        for  all w ∈ Q_next_tmp  do in parallel // пока не закончена phase1
            if  (!acswp(Marked[w], 0, 1) then Q_next[afinc(Q_next_counter)] = w
        end for
    end phase2
    Barrier // общая барьерная синхронизация
end while
```

Рис. 3. Схема распределенного алгоритма *parallel phases* поиска вширь в графе с совмещением фаз

Поскольку вершины из списков смежности могут принадлежать другим узлам, алгоритм разделяется на две фазы. Первая фаза состоит в просмотре списков смежности для вершин из локальных массивов Q и рассылке вершин из этих списков по узлам, которым принадлежат рассылаемые вершины. Вторая фаза заключается в проверке полученных вершин и, возможно, добавлении на следующий уровень Q_{next}.

Фазы могут выполняться последовательно, но эффективнее выполнять их параллельно. Передача вершины с первой фазы на вторую происходит путем непосредственной записи во вспомогательный массив Q_{next_tmp} того узла, которому принадлежит вершина. На этом узле тред второй фазы (проверяющий тред) по счетчику $Q_{next\_tmp\_counter}$ определяет, что пришла новая вершина и проводит проверку. На рис. 3 верхний индекс в имени массива или переменной означает номер узла, на котором находится массив или переменная, имя без индекса означает использование локального массива или переменной. Функция *nodeNumber*(*w*) вычисляет номер узла, которому принадлежит вершина *w*.

Так как между атомарным увеличением счетчика $Q_{next\_tmp\_counter}$ и фактической записью проходит достаточно большое время, то запись и считывание проводятся в режиме *synchronize*. Дело в том, что каждая 64-разрядная ячейка памяти МТП-суперкомпьютера имеет дополнительный теговый *f*/*e*-бит. В обычном режиме *f*/*e*-бит не влияет на работу. В режиме *synchronize* чтение из ячейки со значением *f*/*e*-бита *empty* не выполняется, аппаратура обеспечивает ожидание треда без выборки с него команд. Запись в режиме *synchronize* меняет состояние ячейки на *full*. Когда ждущая команда видит значение *f*/*e*-бита *full*, она меняет состояние обратно на *empty* и возвращает значение. Этот механизм обеспечивает удобную и эффективную синхронизацию тредов. Перед началом работы *f*/*e*-биты ячеек массива Q_{next_tmp} инициализируются значением *empty*.

В алгоритме *parallel phases* на каждом уровне в среднем будет $2k|Q|$ глобальных операций, а также некоторое число глобальных чтений из Edges, не попавших в локальную память узлов, что в сумме значительно меньше по сравнению с алгоритмом *uniform*. Тем не менее, архитектурные особенности МТП-суперкомпьютера позволяют еще улучшить алгоритм поиска вширь, что рассмотрено в следующем разделе.

Распределенный вариант на основе RPC. В системе команд МТП-суперкомпьютера имеется команда Remote-ProcedureCall (RPC) запуска с узла-отправителя на заданном узле-получателе треда с передачей ему нескольких аргументов. Запускаемый тред на узле-получателе выбирается из не-

```
// инициализация
while  sum_{i=1}^{nodesNumber} Q^i_counter ≠ 0  do
    for all vertex ∈ Q  do in parallel
        for all w ∈ E_vertex  do
            RPC(phase2, w, nodeNumber(w))
        end for
    end for
    Barrier // синхронизация посылающих тредов
end while
procedure phase2(w)
    if  (!acswp(Marked[w], 0, 1) then  Q_next[afinc(Q_next_counter)] = w
end phase2
```

Рис. 4. Схема распределенного алгоритма поиска вширь в графе на основе команды RPC

скольких заранее выделенных тредов для обслуживания RPC. Если тред невозможно по какой-то причине запустить, то на узел-отправитель в регистр результата записывается соответствующее значение кода возврата.

На рис. 4 представлен алгоритм RPC, в котором проверяющему треду на целевом узле номер вершины передается с помощью команды RPC. Такой способ позволяет избежать использования массива $Q_{\text{next\_tmp}}$, *f/e*-битов и сократить вдвое число глобальных операций для передачи вершин. Действительно, не требуется атомарного увеличения счетчика для добавления вершины, вершина непосредственно передается на проверку и, в случае необходимости, добавляется на следующий уровень. В остальном алгоритм очень похож на предыдущий.

Число глобальных операций для алгоритма RPC составляет в среднем $k|Q|$ на каждом уровне с добавлением тех же глобальных обращений, не попавших в локальные части Edges. Также если проверяющие треды будут заняты при приходе на узел-получатель команды RPC, RPC программно переповторяется на узле-отправителе, что увеличивает число глобальных операций. Однако число переповторов снижается практически до нуля путем оптимального подбора числа тредов, работающих на первой фазе рассылки смежных вершин, и числа проверяющих тредов. В данном варианте алгоритма число тредов разного типа не меняется в течение времени выполнения алгоритма.

5. Анализ производительности суперкомпьютеров на тесте BFS

На рис. 5 и 6 представлена производительность трех вариантов алгоритма поиска вширь в графе в зависимости от числа узлов для J7-2 и J10-4 соответственно.

Алгоритм *uniform* демонстрирует самую плохую масштабируемость, в частности, для J10-4 при использовании большого числа узлов не удается достичь преимущества по сравнению с производительностью на одном процессоре, что объясняется большим дисбалансом пропускных способностей памяти и сети, приведенных в табл. 1. В алгоритме RPC доля глобальных операций по отношению к локальным меньше, чем в алгоритме *parallel phases*, что обеспечивает алгоритму RPC самую лучшую масштабируемость. Критическим ресурсом при выполнении всех алгоритмов является пропускная способность сети, недостатки реализации алгоритмов *parallel phases* и RPC в использовании сети позволили алгоритму *uniform* обогнать *parallel phases* по производительности при небольшом числе узлов.

В табл. 2 представлены наивысшие значения производительности для задачи поиска вширь в графе, полученные на разных архитектурах. На BlueGene/L обработан наибольший граф из всех известных на октябрь 2008 года, состоящий из $3{,}2 \cdot 10^9$ вершин. Для BlueGene/L используется

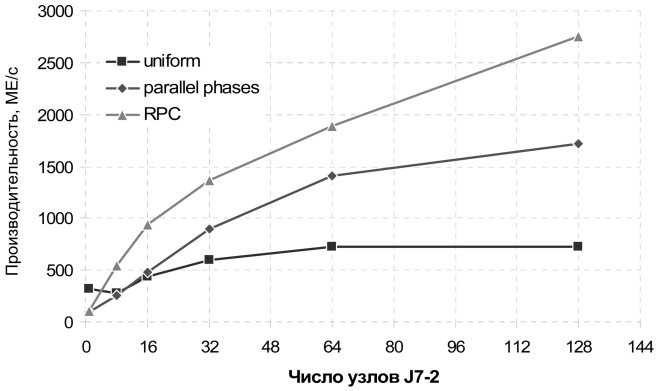


Рис. 5. Производительность разных алгоритмов поиска вширь, $N = 10^5$, $k = 10$, J7-2 (2 ядра по 64 треда)

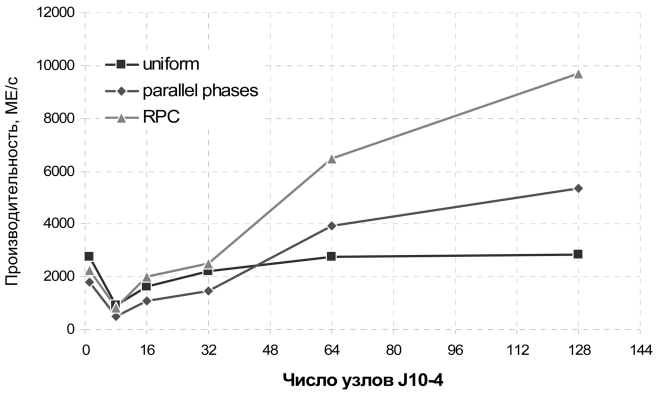


Рис. 6. Производительность разных алгоритмов поиска вширь, $N = 10^6$, $k = 10$, J10-4 (8 ядер по 128 тредов)

Таблица 2

Производительность (МЕ/с) различных систем на задаче поиска вширь для графов Пуассона с разными параметрами

Система	Число вершин в графе N	Производительность, МЕ/с		
		$k = 10$	$k = 50$	$k = 100$
BlueGene/L, 128 узлов	$3{,}2 \cdot 10^9$	50	175	310
BlueGene/L, 256 узлов	$3{,}2 \cdot 10^9$	80	220	450
BlueGene/L, 32 768 узлов	$3{,}2 \cdot 10^9$	7655	—	—
BlueGene/L, 32 768 узлов	$3{,}2 \cdot 10^8$	—	—	11 930
МВС-100*k*, 32 узла	$3{,}2 \cdot 10^6$	228	—	986
МТА-2, 40 узлов	Нет свед.	530	810	900
МТП, J7-2, 1024 узла, RPC	$3{,}2 \cdot 10^6$	21 021	—	—
МТП, J10-4, 1024 узла, RPC	$3{,}2 \cdot 10^6$	55 635	—	—

алгоритм 1D-разбиения [9], в котором описанные две фазы алгоритма *parallel phases* выполняются последовательно. Полученная производительность является наивысшей из всех известных результатов на реальном оборудовании, однако при этом используется очень большое число узлов.

Для современного кластера МВС-100*k* получены хорошие результаты на алгоритме 2D-разбиения [9], впервые реализованного на Charm++ авторами статьи. Отметим результаты для уже старой системы МТА-2. Преимущество МТА-2 заключается в очень простом программировании и хорошем масштабировании: приведенные результаты получены для программы, подобной *uniform* алгоритму.

Ограничения по времени моделирования заставили рассмотреть для МТП-суперкомпьютера граф, меньший графа для BlueGene/L на три порядка, но результаты все равно сравнимы. Дело в том, что оценки числа глобальных операций получены в предположении, что все обращения в распределенные массивы будут глобальными. Однако часть из этих обращений все же попадает в локальную память узлов, и при увеличении размера задачи для того же числа узлов доля локальных обращений только увеличится. Отметим, что результаты для МТП-суперкомпьютера получены всего лишь на 1024 вычислительных узлах.

Заключение

В статье рассмотрены три варианта (*uniform*, *parallel phases* и RPC) организации решения задачи поиска вширь в графе на МТП-суперкомпьютере с глобально-адресуемой памятью. Полученные результаты очень высоки в сравнении с результатами реализации поиска вширь на различных архитектурах и доказывают возможность МТП-суперкомпьютера эффективно решать задачи с интенсивной нерегулярной работой с памятью.

Глобально-адресуемая общая память позволила на языке ассемблера очень просто реализовать алгоритм с неоднородной организацией тредовых вычислений. Возможности архитектуры МТП-суперкомпьютера позволили обеспечить во втором алгоритме *parallel phases* мелкозернистую синхронизацию на каждой ячейке памяти, а в третьем алгоритме RPC — обеспечить локализацию вычислений при данных, минуя память.

Дальнейшие исследования будут посвящены рассмотрению задачи поиска вширь на других типах графов, а также исследованию производительности МТП-суперкомпьютера на других задачах обработки графов.

Авторы статьи выражают благодарность руководителю работы по разработке МТП-суперкомпьютера Л. К. Эйсымонту.

Список литературы

1. **Фролов А., Семенов А., Корж А., Эйсымонт Л.** Программа создания перспективных суперкомпьютеров // Открытые системы. 2007. № 9. С. 21—29.

2. **Bader P., Feo J.** and et al. HPCS SSCA#2 Graph Analysis Benchmark Specifications v1.1. Jan. 2006.

3. **Lumsdaine A., Gregor D., Hendrickson B., Berry J. W.** Challenges in Parallel Graph Processing // Parallel Processing Letters. 2007. V. 17. № 1. С. 5—20.

4. **Bader D., Cong G., Feo J.** On the Architectural Requirements for Efficient Execution of Graph Algorithms // Proceedings of ICPP'05. 2005. Vol. 00. P. 547—556.

5. **Слуцкин А., Эйсымонт Л.** Российский суперкомпьютер с глобально-адресуемой памятью // Открытые системы. 2007. № 9. С. 42—51.

6. **Charm++** Homepage. http://charm.cs.uiuc.edu/.

7. **Villa O., Scarpazza D. P., Petrini F., Peinador J. F.** Challenges in Mapping Graph Exploration Algorithms on Advanced Multi-core Processors // Proceedings of IPDPS. 2007.

8. **Bader D., Madduri K.** Designing Multithreaded Algorithms for Breadth-First Search and st-connectivity on the Cray MTA-2 // Proceedings ICPP, Columbus, OH. 2006.

9. **Yoo A., Chow E., Henderson K., McLendon W., Hendrickson B., Catalyrek U**. A Scalable Distributed Parallel Breadth-First Search Algorithm on Blue-Gene/L // Proceedings of SuperComputing'05, Seattle, WA. 2005. November.

УДК 004.27

А. Е. Мамченко, канд. техн. наук, доц.,
Московский государственный университет
путей сообщения (МИИТ),
e-mail: vss.miit@gmail.com

Образовательно-методологический аспект "принстонской" и "гарвардской" архитектур процессоров вычислительных систем

Широко распространенные архитектуры ЭВМ и однокристальных микропроцессоров и микроконтроллеров — классическая (принстонская, фон-Неймановская) и гарвардская — рассматриваются с точки зрения оперируемых ими адресных пространств памяти, адресами которых идентифицируются команды и обрабатываемые данные. Обсуждается также модифицированная гарвардская архитектура и образование последней из гарвардской в случае одной пары шин адреса и данных.

Ключевые слова: *вычислительная система, процессор, архитектура.*

Введение

Автор давно интересуется одним из употребительных терминов в лексиконе вычислительной техники — "архитектурой". В работе [1] им была предпринята попытка понять, каким смыслом наполняется он в разных источниках. Определены четыре толкования этого термина, один из которых озвучен так:

"1) обозначение общей (глобальной) концепции или совокупности концепций (принципов) построения ЭВМ или вычислительной системы. В рамках этого толкования понятны словосочетания "ЭВМ с традиционной (фон-неймановской) архитектурой", "ЭВМ с нетрадиционной архитектурой", "RICS-архитектура" [6], "объективно-базированная архитектура вычислительной машины" [4], системы "со специальной архитектурой, ориентированной на достижение отказоустойчивости" [7] и т. п. Термин здесь встречается с присоединением слова "макро": "существует стандартная классификация макроархитектур, основанная на соотношении потоков команд, выполняемых системой, и потоков данных [8];"...

Понятно, что анализ широкого спектра архитектур в рамках отдельной статьи невозможен. Поэтому автор ограничивается в ней в основном рассмотрением простейших из них — однопроцессорных систем, или, по классификации Г. Флина, реализующих тип системы "один поток команд — один поток данных" (SISD — Single Instruction/Single Data) [2]. При этом полагается возможным распространение результатов и на процессорные элементы параллельных систем.

Обсуждение вопроса предваряется еще двумя цитатами из работы [1], где "архитектура" есть:

"4) обозначение более или менее точного перечня сведений об ЭВМ, доступных и необходимых при программировании на языке машинных команд"...

"Более того, автор предлагает связывать с таким понятием архитектуры, следующие архитектурные компоненты:

- структура (структурная схема) ЭВМ, системы или микропроцессора, показывающая основные устройства, блоки и узлы, связи и взаимодействия между ними;
- типы, формы представления и форматы данных, обрабатываемых командами ЭВМ — чисел, логических кодов, битов, символьных данных и др.;
- программистская модель ЭВМ, т. е. перечень всех программно-доступных элементов, в которых располагаются команды и данные, и которые адресуются командами ЭВМ — регистров процессора, ячеек памяти, портов ввода-вывода, отдельных триггеров, входов и выходов и т. п.;
- перечень (система) команд и других управляющих слов с указанием их функций, групп, форматов, способов кодирования и используемых в них способов (механизмов) адресации программно-доступных элементов;
- сведения о структурно-логической организации системы прерываний ЭВМ, включающие данные о классах запросов прерываний, допустимых моментах восприятия запросов, процедуре запоминания состояния прерываемой программы и механизме перехода к прерывающей программе, программном управлении масками и приоритетами запросов прерываний и др.;
- сведения о структурно-логической организации подсистемы ввода-вывода ЭВМ, т. е. о способах обмена, адресации средств ввода-вывода и периферийных устройств, управляющих словах, протоколах ввода-вывода и др."

1. Принстонская (фон-Неймановская) архитектура

Известно, что наиболее распространенным в практике принципом организации ЭВМ является принцип программного управления по хранимой в памяти программе (*stored program*), предложенной в 1945 г. Дж. фон Нейманом. По имени Принстонского университета, в котором работали

Дж. Нейман и его коллеги, такая концепция организации получила название **принстонской** архитектуры [3, 4]. Хотя изложение фон-Неймановского принципа содержится во многих литературных источниках, например в работе [5], автору импонирует его формулировка в работе [6]:

1. Информация кодируется в двоичной системе и разделяется на единицы (элементы) информации, называемые *словами*.

2. Разнотипные слова информации различаются по способу использования, а не способами кодирования.

3. Слова информации размещаются в ячейках памяти и идентифицируются номерами ячеек, называемыми *адресами слов*.

4. Алгоритм представляется в форме последовательности управляющих слов, которые определяют наименование операции и слова информации, участвующие в операции, и 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.

Авторы, комментируя эти пункты, специально выделяют, что адрес ячейки, в котором хранится величина или команда, становится *машинным идентификатором* (именем) величины и команды. Неоспоримым достоинством фон-Неймановского принципа программного управления является то, что арифметико-логическому устройству для обработки доступны данные, помещенные в память, в результате чего обеспечивается высокое быстродействие машины.

Тот факт, что в памяти располагаются и команды, и данные (величины), позволяет говорить о наличии в ЭВМ с принстонской архитектурой *одного* множества адресов (идентификаторов), генерируемого процессором ЭВМ — *адресного пространства памяти программ и данных* (АППД), а саму оперативную память называть *памятью программ и данных* (ППД). Автор полагает, что введение понятия "адресное пространство" позволяет рассуждать об архитектурах ЭВМ и систем наиболее корректно. При этом размер такого пространства определяется только разрядностью k формируемого процессором[1] на своей внешней адресной шине адреса (а не механизмами его образования внутри процессора и способами последующей интерпретации):

$$C_{\text{АППД}} = 2^k.$$

[1] Понятие "процессор ЭВМ", как известно, объединило арифметико-логическое устройство ЭВМ и устройство управления ее в одно целое.

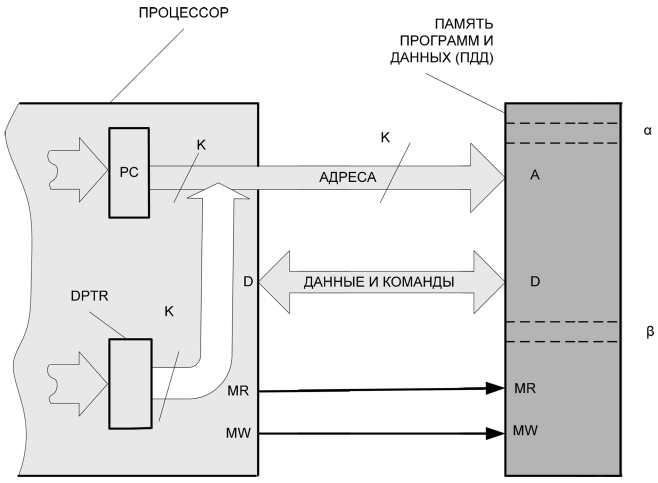


Рис. 1. Взаимодействие процессора и памяти в системе с принстонской архитектурой и двунаправленной шиной данных (A — шина *k*-разрядного адреса, D — шина данных, MR и MW — сигналы "*memory read*" и "*memory write*")

В принстонской архитектуре мы имеем два источника адреса от процессора к памяти: команды адресуются содержимым программного счетчика (PC-*program counter*), а данные — содержимым одного (или одного из нескольких) регистра адреса данных (DPTR-*data pointer register*) (рис. 1). Принципиально оказывается возможным расположение в ячейке с произвольным адресом (α, β на рис. 1) или команды, или слова данных (операнда, промежуточного или конечного результата), а также чтение командой "самой себя"[2].

Наличие в системах с принстонской архитектурой одного АППД определяет исключительно важное свойство таких ЭВМ и систем: возможность модификации (изменения и последующего сохранения) команд, и, главное, обработки одних команд другими как данных. Это сделало реальным автоматизацию программирования, т. е. перевода программ с языка высокого уровня на машинный язык ЭВМ. Именно этим свойством можно объяснить исключительное распространение систем с принстонской архитектурой и закрепление за ней названия традиционной или классической.

Следует обратить внимание на одну "маленькую" нестыковку в принципе фон-Неймана, обходимую в популярных и даже фундаментальных

[2] Конечно, это утверждение не следует понимать буквально:
- при начальной инициализации процессора происходит загрузка PC начальным (стартовым) адресом (например, установка его в "0"), и в ячейке с этим адресом должна располагаться только команда;
- указанная выше команда может быть началом программ инициализации процессора, тестирования памяти, загрузки программ взаимодействия с периферией и т. п.;
- ряд ячеек ППД могут использоваться только для расположения системной информации (векторов прерываний и др.).

источниках: как попадает программа в "пустую" до начала работы ЭВМ память?

Здесь возможны два ответа:

1) "внепрограммно", т. е. есть аппаратура, загружающая в ППД часть программы, делающей возможным последующий ввод и исполнение собственно программы задачи;

2) реализация ППД в виде подсистемы ROM *(read only memory*) и RAM (*random access memory*). При этом программы начальной инициализации всей ЭВМ, включая устройства ввода, находятся в ROM, а программа(ы) решения задачи, "заполняют" RAM под управлением вышеуказанных.

Первое решение имело место в первых ЭВМ 60—70 гг. XX века. Второе решение стало использоваться с появлением персональных компьютеров и является на взгляд автора, наиболее простым.

Для обмена с периферией (ввода/вывода) в системах с принстонской архитектурой необходимы порты ввода и вывода. Здесь возможны:

1) адресация портов ввода/вывода, через которые происходит обмен, адресами адресных пространств ввода и вывода (АПВв и АПВыв). В этом случае в системе команд присутствуют команды ввода и вывода типа IN port и OUT port, а сам ввод/вывод осуществляется с участием тех же шин адреса A (могут участвовать не все его разряды), шины данных и сигналов управления IOR (input/output read) и IOW (input/output write);

2) адресация портов ввода/вывода частью адресов адресного пространства памяти программ и данных, специально выделяемой для этих целей. При этом командами ввода/вывода становятся команды обращения к памяти с адресами из этого диапазона, а при вводе/выводе используются те же шины и сигналы, что и на рис. 1. Такое решение имело место в ЭВМ 70-х годов прошлого века: PDP-11 фирмы DEC и отечественных СМ-3,4 из системы малых ЭВМ (СМ ЭВМ).

Таким образом, введенный ранее термин "адресное пространство" распространим не только на основной ресурс ЭВМ — память, но и на средства ввода/вывода, которые, в соединении с программно-доступными регистрами внутри процессора, определяют программистскую модель машины и системы.

Примером реализации принстонской архитектуры с адресацией портов ввода/вывода адресами собственных адресных пространств могут служить системы на основе универсальных однокристальных микропроцессоров Intel линии X86 (рис. 2). В них шина данных имеет разрядность 8*p* разрядов, (возможно и 9*p*, если имеется контрольный разряд), причем $p = 2$ (*Intel* 8086), $p = 4$ (*Intel* 80486), $p = 8$ (*Pentium*) [9]. Память программ и данных организована в виде p байтовых банков, обращения к которым организуется с помощью

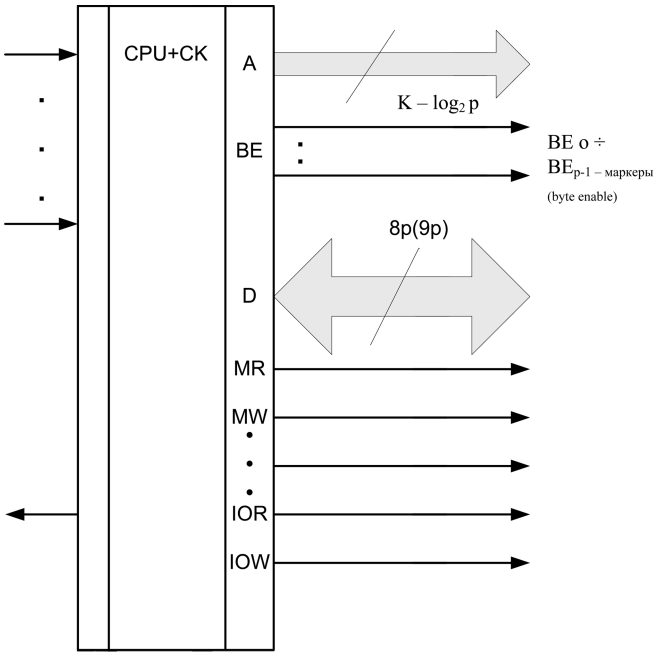


Рис. 2. Шины и сигналы однокристального микропроцессора с принстонской архитектурой и адресацией портов ввода/вывода адресами собственных адресных пространств (СК — системный контроллер)

($K - \log_2 p$)-разрядного адреса и p маркеров BE (*byte enable*). Сигналы MR, MW, IOR и IOW образуются (с помощью системного контроллера CK, являющегося обычно частью чипсета), из вырабатываемых микропроцессорами сигналов R (*read*), W (*write*) и M/IO (*memory/input-output*). При $k = 32$ и $p = 4$ шина адреса имеет 30 разрядов.

Попутно отметим, что в многопроцессорных системах адресами периферии могут выступать не порты, а процессоры ввода/вывода. Так, в ЭВМ систем IBM 360, 370 и отечественных ЭВМ Единой системы (ЕС ЭВМ) адресами в командах ввода/вывода (например, SIO — Start Input/Out put) выступают как раз номера специализированных каналов (процессоров) при выполнении которыми канальных программ и ведется обмен между памятью и периферией.

Таким образом, принстонская архитектура может быть определена как архитектура с **одним** адресным пространством памяти — общим для адресации как команд (программы), так и обрабатываемых данных.

2. Гарвардская архитектура

Интересно, что **гарвардская** архитектура была предложена (в Гарвардском университете, США) ранее принстонской: в конце 30-х годов XX века. И хотя обе архитектуры объединяет, например, использование двоичного кодирования команд и данных, машинными идентификаторами (адреса-

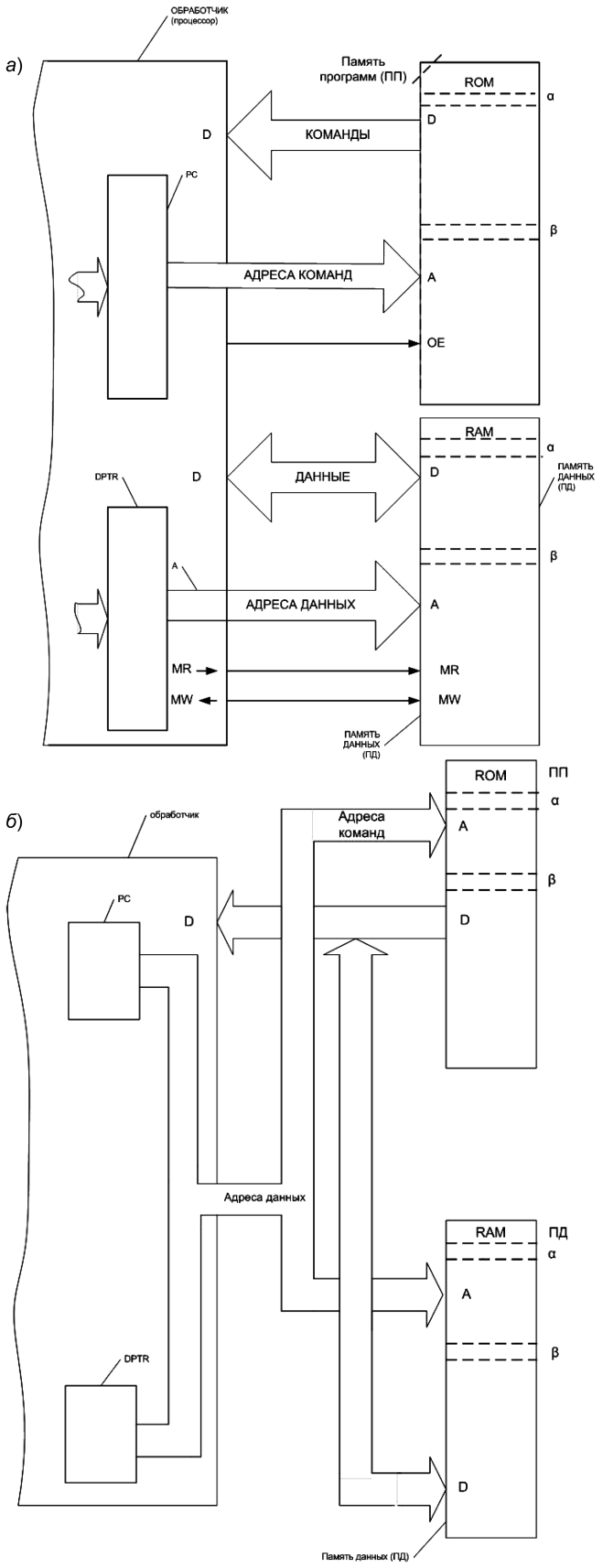


Рис. 3. Взаимодействие процессора и памяти в системе с гарвардской архитектурой при наличии раздельных шин адреса и данных *(а)* и при совмещении тех же шин (*б*)

ми) этих команд и данных выступают множества адресов разных адресных пространств:

- адресного пространства памяти программ (АППП), используемого для идентификации/адресации команд;
- адресного пространства памяти данных (АППД), адресами которого идентифицируются данные.

При этом (рис. 3, *а*) обращение к ячейкам памяти программ, адресуемым содержимым программного счетчика PC, может быть только чтением и осуществляется при действии сигнала PSEN (*program storage enable*), поступающего на вход OE (*out enable*) памяти ROM. В результате функционирование систем с гарвардской архитектурой возможно только по фиксированным в программной памяти программам.

Обращение по адресу из DPTR относится к ячейкам памяти данных и сопровождается одним из сигналов MR или MW. Сразу отметим, что адресами адресного пространства памяти данных могут быть адресованы также порты ввода/вывода периферийных устройств. При равномощности адресных пространств АППП и АППД адреса α и β обнаруживаются в обоих множествах адресов. Таким образом, логическое и физическое разделение памяти программ и памяти данных является принципиальным отличием систем с гарвардской архитектурой от систем с принстонской.

Из рис. 3, *а* видно, что реализация систем с гарвардской архитектурой при наличии двух пар шин позволяет обеспечить потенциально заложенную в этой архитектуре более высокую скорость выполнения программы: осуществлять выборку следующей команды одновременно с выполнением текущей. Если такой механизм параллелизма в обработчике не предусмотрен, структура системы с гарвардской архитектурой может содержать общую шину адреса и двунаправленную шину данных (рис. 3, *б*). Поэтому встречающиеся в литературе определения архитектуры как "двухшинной" или с "раздельными шинами данных и команд" представляются не совсем корректными. Нельзя также согласиться с утверждением, что наличие отдельных кэш-памяти команд и кэш-памяти данных в принстонской архитектуре превращает последнюю в гарвардскую [7], так как обе кэш-памяти реализуют "теневой", не отраженный в системе команд, механизм функционирования машины и являются более поздним "изобретением" вычислительной техники.

Отметим, наконец, что размеры адресных пространств памяти во всех архитектурах определяют максимальные объемы памяти, к которым обращается процессор (микропроцессор, микроконтроллер). В реальных системах имеет место память меньшего объема и используются различные

способы распределения адресов адресных пространств между ячейками или, иначе, назначения адресов пространств ячейкам памяти [11]. Естественно, в персональных компьютерах и системах обработки данных применяется полноразрядная адресация с назначением каждой ячейке только одного адреса из адресного пространства. В более простых системах возможна неполная дешифрация адреса, т. е. назначение каждой ячейке группы смежных адресов пространства, что приводит к более простой аппаратной реализации систем, и даже бездешифраторная адресация — применительно к портам ввода/вывода при адресации последних адресами собственных адресных пространств.

Широкое распространение гарвардской архитектуры начинается с 70-х годов XX века, когда появляется широкий спектр однокристальных микроконтроллеров (однокристальных микроЭВМ) для построения управляющих микропроцессорных систем для АСУ ТП низового уровня. Хорошо зарекомендовавшими себя такими микроЭВМ могут считаться однокристальные микроконтроллеры (ОМК) семейства MCS-51 фирмы *Intel*. Они имеют в своем составе резидентную память (РПД) из 256 байтовых ячеек, некоторые из которых способны выполнять спецфункции, функции портов ввода/вывода и быть элементами внутренней периферии. Микроконтроллерами адресуется внешняя память программ объемом 64 Кбайт и память данных такого же объема. Часть адресов АППД памяти может быть использована для адресации дополнительных портов ввода/вывода. Базовый кристалл семейства — *Intel* 8051 — имеет резидентную (внутрикристальную РПП) память программ объемом 4 Кбайт, адресуемую начальными адресами АППП [8]. На ОМК семейства можно строить разнообразные, по сложности и возможностям, управляющие микросистемы [10]. На рис. 4 приведены некоторые варианты таких систем. Инверсии над сигналами PSEN, MR, MW отражают реалии чипов CPU.

В ОМК этого семейства возможна адресация разнородных по функциям программно-доступных элементов адресами одного адресного пространства (резидентной памяти данных). В то же время некоторые из таких элементов могут адресоваться номерами разных адресных пространств. Это дает возможность выбирать при программировании разные по формату и времени исполнения команды из системы команд ЭВМ и создавать эффективные либо по быстродействию, либо по занимаемому объему памяти программы.

3. Модифицированная гарвардская архитектура

Существует также структура, соединяющая в себе свойства принстонской и гарвардской архитектур. Она носит название **модифицированной гарвардской** архитектуры и оперирует с тремя адресными пространствами:

1) адресным пространством памяти программ (АППП);

2) адресным пространством памяти программ и данных (АППД);

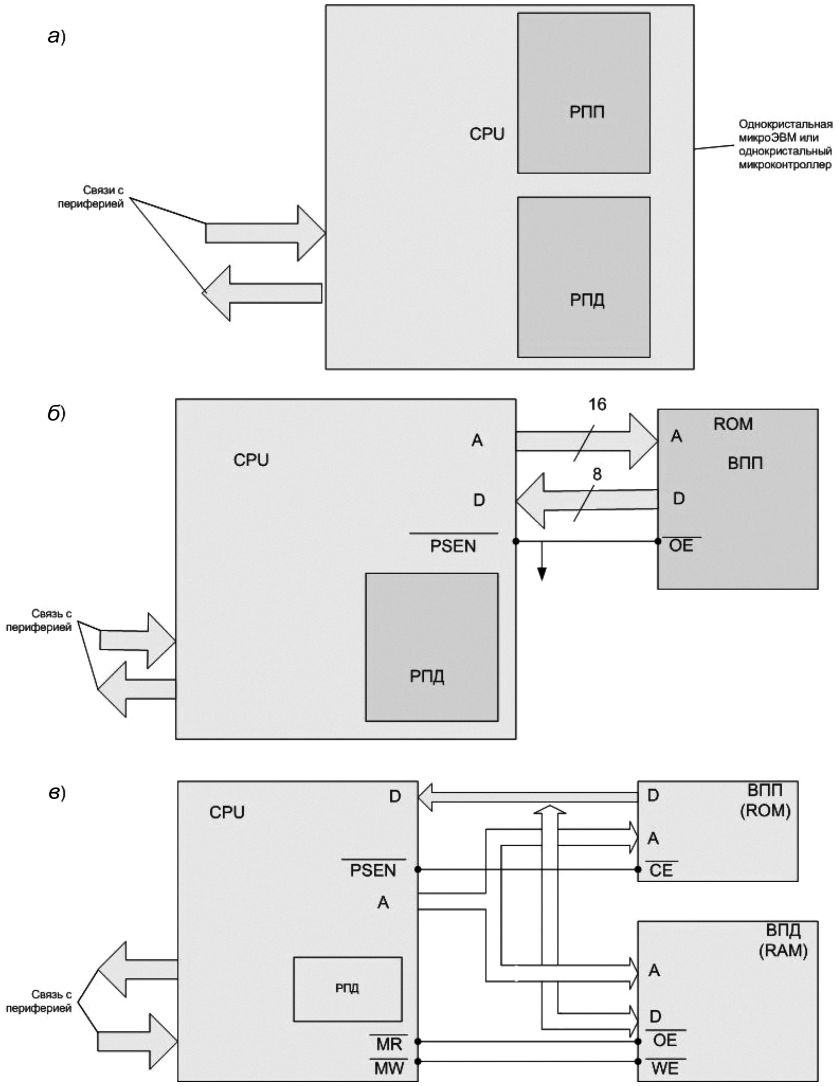


Рис. 4. Некоторые варианты реализации систем с гарвардской архитектурой: с резидентной (внутрикристальной) памятью программ и памятью данных (*а*), резидентной памятью данных и внешней памятью программ (*б*), внешними памятью программ и памятью данных и шинами адреса и данных (*в*)

3) адресным пространством памяти данных (АППД).

Понятно, что в памяти, адресуемой адресами первого из них, могут располагаться только команды; источником адреса является программный счетчик PC, а в качестве самой памяти используется ПЗУ (ROM). Адресами АППП адресуется память, в которой могут располагаться как команды, так и данные, а источниками адреса выступают содержимые и PC, и DPTR. Наконец, пространство АППД используется для адресации памяти типа RAM, где располагаются только данные. Что касается адресации портов ввода/вывода, то они могут адресоваться частью адресов пространств АППП и АППД.

Модифицированная гарвардская архитектура оказывается удобной для учебных микропроцессорных систем, служащих для изучения микроконтроллеров с гарвардской архитектурой и систем на их основе. В этом случае в памяти программ располагается системное программное обеспечение (МОНИТОР), под управлением которого и с использованием средств ввода и индикации в память программ и данных загружается пользовательская программа. Память данных в таких системах является и пользовательским, и системным ресурсом.

Гарвардскую архитектуру микроконтроллера с внешними устройствами памяти и общими шинами адреса и данных можно превратить в модифицированную гарвардскую следующим образом [10]:

- начальную часть адресов АППП (если стартовый адрес нулевой) используют для адресации постоянной памяти программ, организуя обращение к ней с помощью сигнала PSEN;
- другую часть этого адресного пространства и точно такую же часть адресного пространства памяти данных выделяют для образования адресного пространства памяти программ и данных, т. е. АППП. Сигнал чтения OE для этой памяти образуется из сигналов PSEN и MR, в качестве сигнала записи WE (*write enable*) используется WR;
- для адресации ячеек памяти данных (и внешних портов ввода/вывода) используется незадействованное (после совмещения на предыдущем шаге) подмножество адресов АППД.

Понятно, что выделение групп адресов отразится в сигналах CS (*chip select*), необходимых для функционирования устройств памяти в модифицированной гарвардской архитектуре. При полном объединении АППП и АППД гарвардская архитектура превращается в принстонскую.

Отдельно следует остановиться на CISC (*complete instruction set computer*)-архитектуре и RICS (*reduced instruction set computer*)-архитектуре. Свойствами полноты и сокращенности системы команд как машинного языка ЭВМ могут обладать процессоры (микропроцессоры) как с принстонской, так и с гарвардской архитектурами, и в этом смысле CISC- и RISC-архитектуры правильнее считать подархитектурами, хотя и важными с точки зрения достижения высоких показателей производительности [9].

В заключение отметим, что большое разнообразие типов современных микропроцессоров (универсальные, сигнальные, медийные, микроконтроллеры, транспьютеры и др.) и широкая номенклатура их в каждом из типов делают их обоснованную классификацию по архитектурам вопросом отдельного исследования.

Список литературы

1. **Мамченко А. Е.** О термине "архитектура" в вычислительной технике // Вестник МИИТа. 2000. Вып. 3.
2. **Барский А. Б.** Параллельные информационные технологии: Учебное пособие. М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007.
3. **First** Draft of a Report on the EDVAC by John von Neumann // Moore School of Electrical Engineering University of Pennsylvania. 1945. June 30.
4. **Беркс А., Голдстейн Г., Нейман Дж.** Предварительное рассмотрение логической конструкции электронного вычислительного устройства // Кибернетический сборник. Вып. 9. М.: Мир, 1964. С. 7—67.
5. **Норенков И. П.** Краткая история вычислительной техники и информационных технологий // Приложение к журналу "Информационные технологии". 2005. № 9.
6. **Майоров С. А., Новиков Г. И.** Структура электронных вычислительных машин. Л.: Машиностроение. Ленингр. Отделение, 1979.
7. **Таненбаум Э.** Архитектура компьютера. 4-е изд. СПб: Питер, 2003.
8. **Сташин В. В., Урусов А. В., Мологонцева О. Ф.** Проектирование цифровых устройств на однокристальных микроконтроллерах. М.: Энергоатомиздат, 1990.
9. **Корнеев В. В., Киселев А. В.** Современные микропроцессоры. М.: Нолидж, 1998.
10. **Мамченко А. Е.** Микропроцессоры в управляющих микросистемах: Конспект раздела лекций по дисциплинам цикла "Микропроцессоры и микропроцессорные системы". М.: МИИТ, 2002.
11. **Мамченко А. Е.** Адресация внешних программно-доступных элементов микропроцессоров и распределение их адресов при проектировании микросистем. Тезисы докладов 2-й Международной научно-технической конференции "Актуальные проблемы развития железнодорожного транспорта". М.: Изд. МИИТ, 1996. Т. 2. С. 38—39.

УДК 004.315

А. И. Грушин, канд. техн. наук, вед. науч. сотр.,
М. Л. Ремизов, инж.-констр.,
А. В. Ростовцев, инж.-констр.,
ИТМ и ВТ им. С. А. Лебедева РАН, г. Москва,
e-mail: aigrushin@ipmce.ru,
Д. Д. Николаев, студент,
Чинь Куанг Киен, студент,
Московский физико-технический институт
(государственный университет)

Высокопроизводительное устройство для обработки радиолокационной информации

Одной из характерных задач обработки радиолокационной информации является вычисление комплексной матрицы в режиме реального времени. В работе рассмотрена реализация алгоритма рекурсивного вычисления комплексной матрицы 64 × 64.

Ключевые слова: *вычислительное устройство, матрица, комплексное умножение с накоплением, числа с плавающей запятой, ПЛИС.*

Введение

В работе [1] показано, что при использовании адаптивного алгоритма выделения сигналов на фоне интенсивных помех с произвольными законами временной модуляции, наиболее трудоемкой в вычислительном отношении является процедура вычисления в режиме реального времени комплексной обратной матрицы $\mathbf{R}_n^{-1}$, зависящей от выборки векторов $\boldsymbol{y}_n$ и константы μ:

$$\mathbf{R}_0^{-1} = \mathbf{I};\ \mathbf{R}_n^{-1} = \mathbf{R}_{n-1}^{-1} - \frac{1}{\mu + \tilde{\boldsymbol{y}}_n \boldsymbol{z}_n} \boldsymbol{z}_n \tilde{\boldsymbol{z}}_n,$$
$$\boldsymbol{z}_n = \mathbf{R}_{n-1}^{-1} \boldsymbol{y}_n,\ n = 1, 2, \ldots, 128, \qquad (1)$$

где $\mathbf{R}_n^{-1}$ — комплексная матрица размерностью 64 × 64; $\mathbf{I}$ — единичная матрица; μ — целая положительная константа; $\boldsymbol{y}_n$ — 64-мерный комплексный вектор; коэффициенты действительной и мнимой частей компонента — 12-разрядные целые числа со знаком; $\tilde{}$ — обозначает комплексное сопряжение и транспонирование; $N = 128$ — число векторов $\boldsymbol{y}_n$ в одной выборке.

Значение матрицы $\mathbf{R}^{-1}$ вычисляется на выборке из 128 векторов $\boldsymbol{y}_n$, затем она передается в другое устройство для принятия решения об обнаружении полезного сигнала. За 5 с необходимо выполнить вычисление матрицы для 576 различных выборок векторов с тремя значениями μ. Вычисления программным способом на компьютере (Core2Duo, 2,66 ГГц, 1 Гбайт) занимают более 43 мин, т. е. не обеспечивают требуемую скорость, поэтому принято решение реализовать вычисления аппаратно с созданием прототипа на ПЛИС.

Анализ формулы, входных данных и результатов моделирования позволил установить, что:

- матрица $\mathbf{R}_n^{-1}$ является эрмитовой;
- $\tilde{\boldsymbol{y}}_n \boldsymbol{z}_n = \tilde{\boldsymbol{y}}_n \mathbf{R}_{n-1}^{-1} \boldsymbol{y}_n$ — положительная действительная величина;
- вычисления в формате *single* чисел с плавающей запятой [2] обеспечивают необходимую точность и диапазон вычислений.

1. Вычислительное устройство

1.1. Схема вычисления и формат чисел

Для вычисления нового значения матрицы нужно провести 128 итераций по формуле (1). Итерацию разделим на этапы (табл. 1).

Здесь MAC — умножение с накоплением (*multiply and accumulate*); DIV — деление; MUL — умножение.

Операции DIV и MUL выполняются над действительными числами, а MAC выполняется над комплексными числами.

Входные данные можно представить числами с фиксированной запятой, так как μ — целое число, а коэффициенты действительной и мнимой частей компоненты вектора $\boldsymbol{y}_n$ — это 12-разрядные целые числа со знаком.

Моделирование показало, что 62-разрядные числа с фиксированной запятой обеспечивают необходимый диапазон вычислений и точность до 10-го двоичного знака по сравнению с вычислениями программным способом в формате *single*. Производительность вычислительного устройства при этом получается меньше требуемой. Использование чисел с плавающей запятой формата *single* позволяет автоматически решить проблему диапазона представления чисел и точности вычислений.

В табл. 2 приведено сравнение реализации вычислителя с использованием чисел с фиксирован-

Таблица 1

Этапы вычислений

	Этап	Операции и объем вычислений
1	$\tilde{\boldsymbol{z}}_n = \mathbf{R}_{n-1}^{-1} \mathbf{y}_n$	64 × 64 MAC
2	$\alpha = \mu + \tilde{\boldsymbol{y}}_n \tilde{\boldsymbol{z}}_n$	1 × 64 MAC
3	$k = \frac{1}{\alpha}$	1 DIV
4	$w_n = -k\tilde{\boldsymbol{z}}_n$	64 MUL
5	$\mathbf{R}_n^{-1} = \mathbf{R}_{n-1}^{-1} - w_n \tilde{\boldsymbol{z}}_n$	64 × 64 MAC

Таблица 2

Два варианта реализации

Характеристика	Фиксированная запятая 62 разряда	Плавающая запятая *single*
Используемая площадь кристалла, %	12	55
Рабочая частота, МГц	300	200—250
Время вычисления обратной матрицы для одного μ, мс	5,6	1
Трудозатраты на разработку вычислителя, чел.-мес.	~6	~15

ной запятой и чисел с плавающей запятой. Первая реализация использует последовательные умножители (экономные, но медленные). Во второй реализации используются более быстрые параллельные умножители, требующие большого объема оборудования. Реализация с использованием чисел с фиксированной запятой предполагает более простые алгоритмы обработки чисел, меньшие трудозатраты на разработку, меньшую площадь кристалла, но требование по производительности повлекло за собой выбор чисел с плавающей запятой.

Исследование алгоритма вычисления обратной матрицы показало, что для обеспечения необходимого диапазона вычислений достаточно 7-разрядного порядка, поэтому в работе предложен формат чисел с плавающей запятой, в котором порядок занимает 7 разрядов, а скрытый бит мантиссы представлен в явном виде (рис. 1). Из четырех режимов округления, требуемых стандартом на двоичную арифметику с плавающей запятой, в устройстве поддерживается режим округления к ближайшему значению. Также не реализованы NaN (not a number — не число) и бесконечности, нули со знаком и денормализованные числа, в случае появления которых используется режим *flush to zero*. Все это упростило реализацию, повысило быстродействие и сэкономило оборудование, причем результат совпадает с вычисленным программным способом в формате *single* с соблюдением всех требований стандарта.

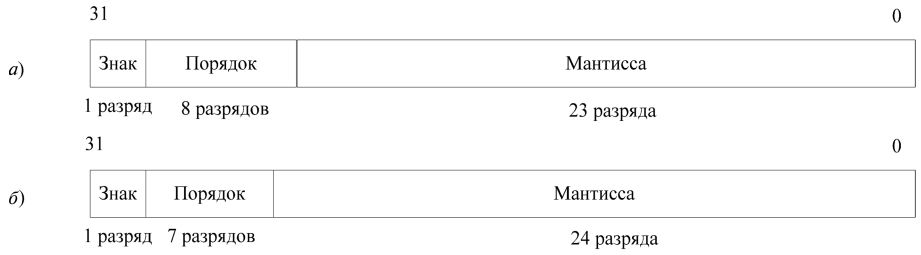
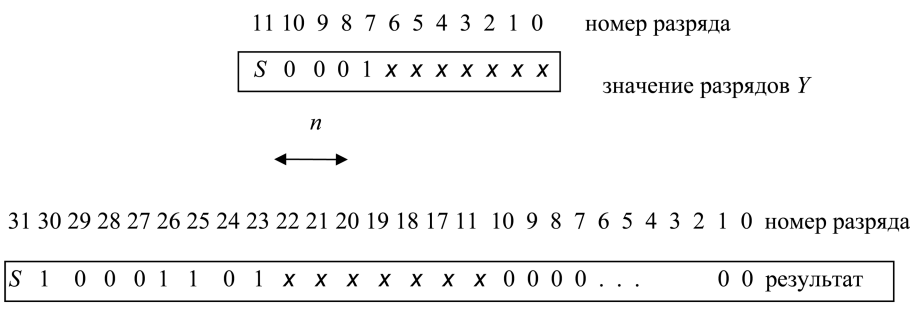


Рис. 1. Форматы чисел с плавающей запятой:
а — формат *single* IEEE Standard 754; *б* — предложенный формат



Здесь S – знак целого числа,
x – значение разряда для определения порядка несущественно,
n – количество старших нулевых разрядов, в примере $n = 3$.

Рис. 2. Пример преобразования целого числа в вещественное

1.2. Основные узлы вычислительного устройства

В табл. 1 приведены операции, необходимые для вычислений по формуле (1). Они реализуются узлами, описанными ниже.

Узел преобразования целого числа в вещественное CI2F. На вход узла подается 12-разрядное целое число Y в прямом коде со знаком (рис. 2). На выходе получается вещественное число, содержащее знак, 7-разрядный смещенный порядок и 24-разрядную мантиссу. Узел является чисто комбинационным. Несмещенный порядок вещественного числа равен $(10 - n)_{10}$, смещенный порядок равен $(10 - n + 63)_{10}$. Мантисса вещественного числа получается в результате сдвига целого числа влево до тех пор, пока старшая "1" исходного числа не окажется в позиции скрытого бита вещественного числа. Мантисса вещественного числа содержит 24 разряда, из которых только 11 старших разрядов могут быть ненулевыми.

Узел комплексного умножения с накоплением MAC. Узел вычисляет следующую функцию (рис. 3):

$$AC - BD + E + \\ + (AD + BC + F)i,$$

где $A + Bi$, $C + Di$, $E + Fi$ — комплексные операнды, причем слагаемое $E + Fi$ может быть результатом предыдущей операции (режим накопления) или подаваться извне. Коэффициенты действительной и мнимой частей операндов — числа с плавающей запятой.

Сначала коммутатор MUX1 пропускает C во входной регистр. Узел умножения чисел с плавающей запятой FPMUL1

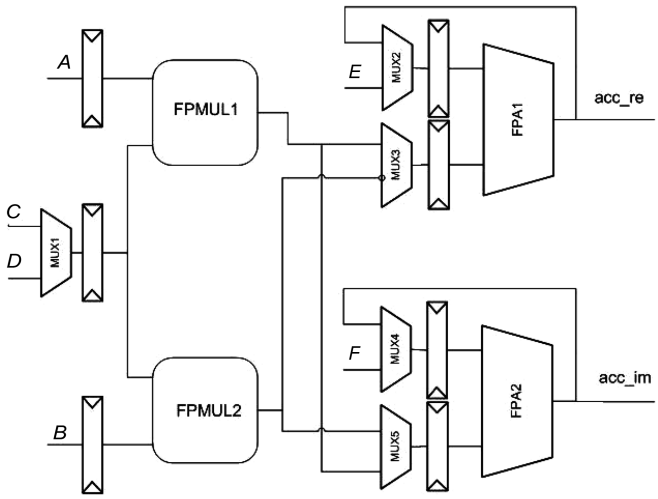


Рис. 3. Структурная схема MAC

вычисляет произведение AC, а узел FPMUL2 — произведение BC. Произведение AC проходит через коммутатор MUX3 на вход узла сложения чисел с плавающей запятой FPA1, на второй вход которого через коммутатор MUX2 проходит E. Произведение BC через коммутатор MUX5 попадает на вход FPA2, где складывается с F.

Затем через MUX1 в узлы умножения проходит D, в FPMUL1 вычисляется произведение AD, в FPMUL2 вычисляется произведение BD. Произведение AD через MUX5 попадает в FPA2, где складывается с $BC + F$, которое через коммутатор MUX4 подается с выхода acc_re. Произведение BD через MUX3 попадает в FPA1, где вычитается из $AC + E$, которое через коммутатор MUX2 подается с выхода acc_im.

В режиме накопления через внешние коммутаторы на входы E и F подаются коэффициенты действительной и мнимой частей вычисленного ранее комплексного результата.

Таким образом, выход acc_re $= AC - BD + E$, acc_im $= AD + BC + F$.

Время выполнения операции MAC 14 тактов, новые операнды на вход MAC можно подавать каждые 8 тактов.

Узел комплексного умножения с накоплением MACR. Узел MACR (рис. 4) представляет собой

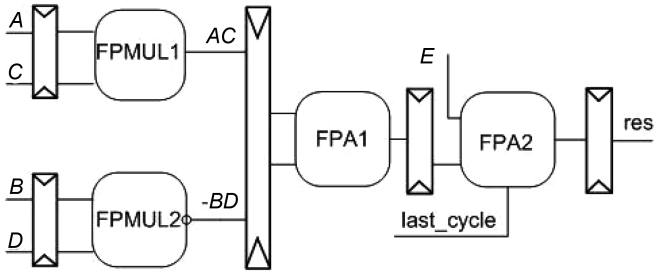


Рис. 4. Структурная схема MACR

комплексный плавающий умножитель с накоплением. На вход узла поступают два комплексных числа $A + Bi$ и $C + Di$ и действительное число E, являющееся результатом предыдущих вычислений или константой, приходящей извне узла. Результат является действительным положительным числом с плавающей запятой res $= AC - BD + E$. Мантиссы чисел A и B содержат 24 разряда, а мантиссы чисел C и D — 11 разрядов.

MACR состоит из двух узлов умножения чисел с плавающей запятой FPMUL1 и FPMUL2 и двух узлов сложения чисел с плавающей запятой FPA1 и FPA2. Для повышения точности при умножении формируется полное 35-разрядное произведение AC, из которого затем вычитается 35-разрядное произведение BD, после чего проводится округление к ближайшему до 35 разрядов. Далее к разности произведений прибавляется E, и сумма округляется до 35 разрядов. Когда сигнал last_cycle = 1, происходит округление до 24 разрядов, т. е. большего из форматов исходных чисел. Время выполнения операции MACR—12 тактов, новые операнды на вход MACR можно подавать каждые 4 такта.

Узел вычисления обратной величины Recipr. Узел использует алгоритм Ньютона—Рафсона [3], итерации проводятся по формуле

$$R_i = R_{i-1}(2 - AR_{i-1}),$$

где R_{i-1} предыдущее приближение; R_i — следующее приближение; A — мантисса нормализованного числа, обратная величина которого вычисляется.

В качестве начального приближения R_0 используется значение из таблицы, хранящейся в постоянной памяти ROM (рис. 5), которое выбирается по шести старшим разрядам дробной части мантиссы числа, подающегося на вход узла. Коммутатор MUX1 пропускает множимое R_0, MUX2 пропускает A, в умножителе MUL1 в следующем такте вычисляется произведение $R_0A[23:0]$. Разность $2 - AR_0$ можно считать дополнительным кодом произведения AR_0, поэтому 48-разрядное произведение инвертируется и через MUX3 попадает в инкрементор, где к нему прибавляется 1.

Второе умножение первой итерации выполняется сразу на двух умножителях MUL1 и MUL2, два произведения с соответствующим смещением складываются в сумматоре Adder. Произведение $R_0(2 - AR_0)$ — это 72-разрядное число, которое округляется к ближайшему до 24 разрядов с помощью инкрементора. Таким образом, в результате первой итерации мы получаем новое приближение мантиссы результата R_1. Две следующие итерации выполняются аналогичным образом.

Мантисса результата нормализуется на один разряд влево, поэтому порядок результата в соответст-

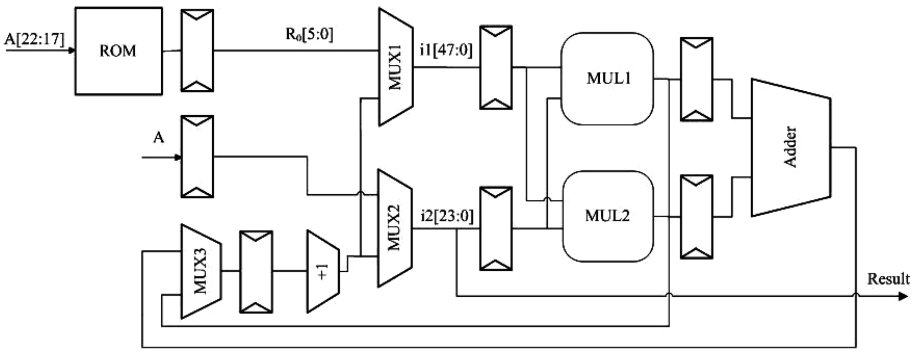


Рис. 5. Структурная схема узла вычисления обратной величины *Recipr*

вии с принятым форматом данных вычисляется по формуле Exp_out[6 : 0] = 125 − Exp_in[6 : 0].

Это соотношение верно за исключением случая, когда скрытый бит равен единице, а дробная часть нулевая. В этом случае нормализации не требуется, и порядок результата определяется следующим образом:

$$\text{Exp\_out}[6:0] = 126 - \text{Exp\_in}[6:0].$$

Устройство выдает результат через 19 тактов после прихода операнда.

1.3. Структура вычислительного устройства

На рис. 6 изображена структурная схема специализированного вычислительного устройства. Основной объем вычислений проводится в блоке из 64 узлов комплексного умножения с накоплением MAC1 − MAC64, каждый из которых выполняет вычисление следующей функции:

$$AC - BD + E + (AD + BC + F)i,$$

где $A + Bi$, $C + Di$, $E + Fi$ — комплексные операнды, причем слагаемое $E + Fi$ может быть результатом предыдущей операции (режим накопления) или подаваться извне. Этот блок позволяет быстро умножать матрицу на вектор, вектор на скаляр, строку на столбец, а также складывать матрицы, т. е. выполнять действия, предусмотренные табл. 1.

Блок комплексных MAC одновременно совершает вычисления над всеми компонентами одного столбца матрицы $\mathbf{R}_n^{-1}$, что обусловливает орга-

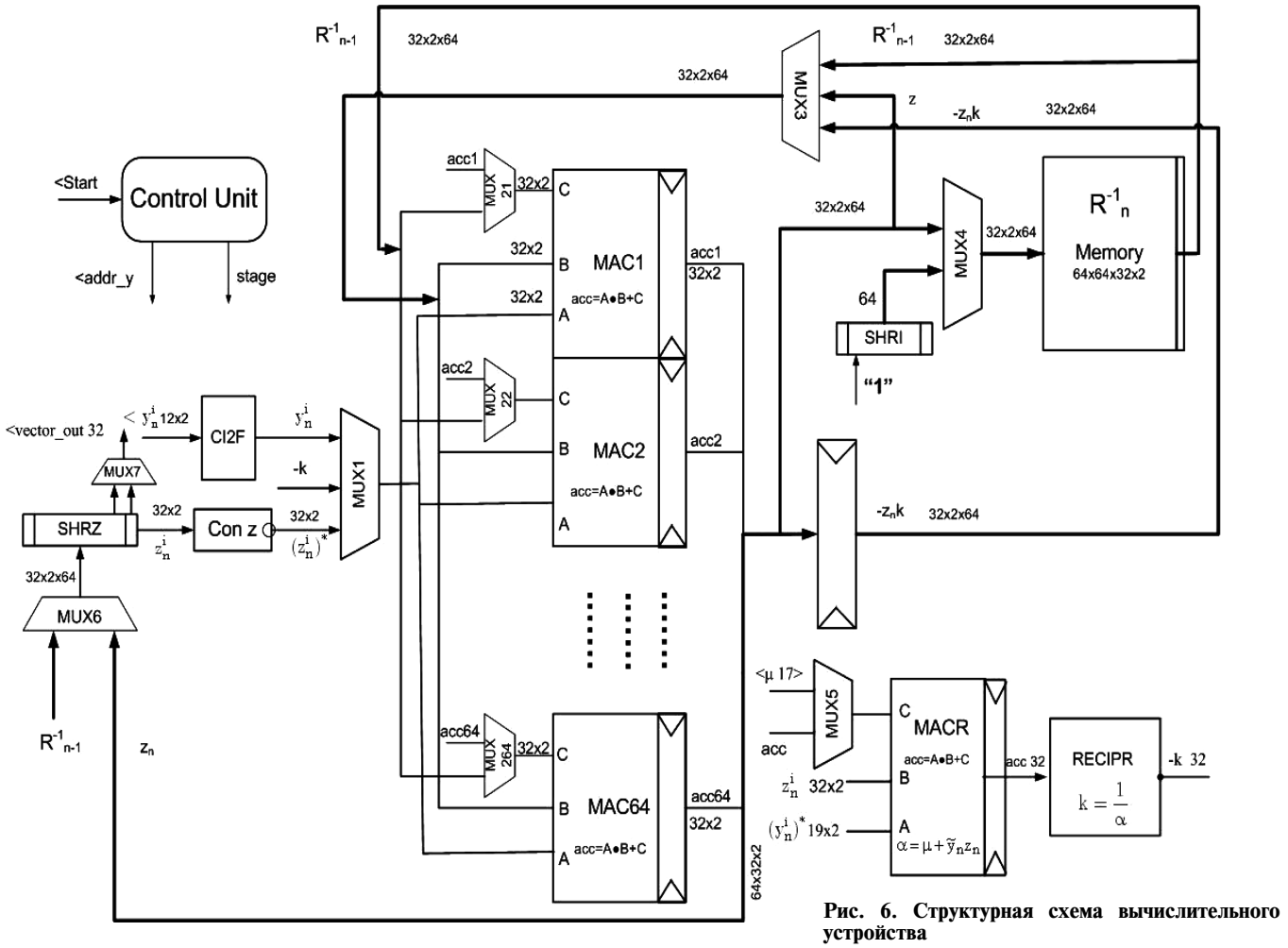


Рис. 6. Структурная схема вычислительного устройства

низацию памяти для хранения матрицы $\mathbf{R}_n^{-1}$. Это память с глубиной 64 (число столбцов) и разрядностью шины данных 64 × 2 × 32, где 64 — количество элементов в столбце; 2 — две компоненты комплексного элемента; 32 — разрядность используемых данных.

В состав устройства также входят: узел преобразования целого в вещественное CI2F, узел MACR (комплексное умножение с накоплением, вырабатывающее только коэффициент действительной части), узел вычисления обратной величины, узел управления Control Unit и другие узлы.

1.4. Интерфейс вычислительного устройства

Входной сигнал *Start* запускает работу устройства. Параметр регуляризации μ определяет возможность различать близко находящиеся цели, от него зависит точность вычислений. Действительная и мнимая компоненты вектора $\boldsymbol{y}_n$ получаются после аналого-цифрового преобразования результата измерений с фазированной антенной решетки.

Адрес addr_y, формируемый в устройстве, используется для чтения компоненты вектора $\boldsymbol{y}_n$ из внешней памяти, младшие 6 разрядов определяют номер компоненты, старшие 7 разрядов определяют номер вектора. В памяти хранятся 128 векторов выборки.

32-разрядный выход vector_out попеременно выдает коэффициент действительной и мнимой частей элемента матрицы. Он используется для передачи вычисленной матрицы во внешнюю память для дальнейшей обработки.

Значимость записи we определяет моменты, в которые происходит запись элементов матрицы во внешнюю память.

1.5. Описание работы вычислителя

Вычисления начинаются с подачи сигнала *Start*, который вырабатывается один раз для обсчета всех 128 векторов $\boldsymbol{y}_n$ выборки. По сигналу *Start* обнуляются счетчики узла управления, и устройство начинает работать в соответствии с временной диаграммой (рис. 7).

Фаза Load_I (начальная загрузка).

В память обратной матрицы $\mathbf{R}_n^{-1}$ записывается вещественная единичная матрица I. Для этого на соответствующий вход коммутатора MUX4 подается коэффициент мнимой части 0 (32 нуля), коэффициент действительной части — вещественная единица (2 нуля, 7 единиц, 23 нуля). В течение 64 тактов прописываются все 64 столбца памяти. В первом столбце вещественная единица записывается в первый (верхний) элемент, во втором столбце — во второй элемент и так далее. Для записи семи единиц в элемент матрицы в 64-разрядном сдвиговом регистре SHRI в каждом такте записи в память единица сдвигается на один разряд, тем самым за 64 такта пробегает от 1-го до 64-го разряда регистра. Выход сдвигового регистра соединен с семью разрядами коммутатора действительной части, которые надо установить в единицу.

Фаза MAC1 (вычисление вектора $\boldsymbol{z}_n$).

Длительность фазы — 64 цикла работы MAC. Коммутатор MUX1 пропускает компоненту $\boldsymbol{y}_n$, которая поступает в MAC1—MAC64 на вход множителя, MUX2 при подаче нулевой компоненты вектора $\boldsymbol{y}_n$ пропускает 0, затем *acc* (результат с выхода MAC), MUX3 — столбец из памяти матрицы $\mathbf{R}_n^{-1}$. Каждый цикл работы MAC1 меняются младшие 6 разрядов адреса чтения $\boldsymbol{y}_n$ addr_y[5 : 0]. За 64 цикла работы блок MAC вырабатывает 64 компоненты вектора $\boldsymbol{z}_n$, которые параллельно принимаются в сдвиговый регистр SHRZ, где преобразуются в последовательный код во время фазы MACR.

Фаза MACR (вычисление знаменателя α). Длительность фазы — 64 цикла работы MACR.

На вход множителя поступает 19 разрядов компоненты $(\boldsymbol{y}_n^i)^*$ (знак, 7 разрядов порядка, 11 старших разрядов мантиссы) с выхода преобразователя целого числа в вещественное CI2F, на вход множимого поступает компонента вектора $\boldsymbol{z}_n$ из сдвигового регистра SHRZ. Через коммутатор MUX5 на вход слагаемого в начальный момент времени поступает константа μ с входа спецвычислителя, затем через этот коммутатор проходит *acc* (результат предыдущей операции MACR). Каждый цикл работы MACR меняются младшие 6 разрядов адреса чтения $\boldsymbol{y}_n$ и происходит сдвиг на одну компоненту в регистре SHRZ. В конце фазы на выходном регистре находится величина α, которая в следующей фазе *Recipr* используется для вычисления обратной величины.

Фаза Recipr (вычисление обратной величины). За три итерации вычисляется обратная величина 1/α, длительность фазы — 19 тактов.

Фаза MAC2 (вычисление произведения $-k \cdot \boldsymbol{z}_n$). Длительность фазы 1 — цикл работы MAC.

С выхода узла *Recipr* через коммутатор MUX1 на вход множителя MAC1—MAC64 поступает величина $-k$. На вход множимого через коммутатор MUX3 поступают компоненты вектора $\boldsymbol{z}_n$, на вход

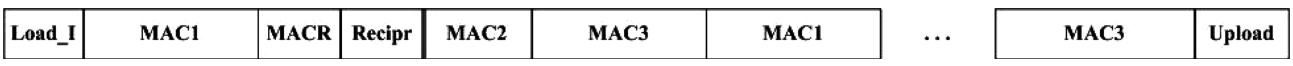


Рис. 7. Временная диаграмма работы вычислительного устройства

слагаемого через коммутатор MUX2 подается 0. Произведение записывается в регистр RgZK.

Фаза MAC3 (вычисление нового значения матрицы $\mathbf{R}_n^{-1}$). Длительность фазы — 64 цикла работы MAC.

Через коммутатор MUX1 на вход множителя MAC1—MAC64 поступает компонента вектора z_n^*, она получается изменением знака мнимой части компоненты вектора z_n на противоположный в узле комплексного сопряжения ConZ. На входы множимого MAC1—MAC64 через коммутатор MUX3 поступают компоненты вектора $-z_n k$. На вход слагаемого MAC1—MAC64 через коммутатор MUX2 из памяти матриц поступает столбец старого значения матрицы $\mathbf{R}_{n-1}^{-1}$. На выходе MAC1—MAC64 получается новое значение столбца матрицы, которое записывается в память матрицы через коммутатор MUX4. Каждый цикл работы MAC в память матрицы записывается новое значение столбца, в сдвиговом регистре SHRZ происходит сдвиг на одну компоненту вектора z_n (тем самым в младших 64 разрядах регистра оказывается следующая компонента вектора z_n), из памяти считывается новый столбец. За 64 цикла работы MAC в память записывается новое значение матрицы. По завершении фазы MAC3 значение старших семи разрядов адреса чтения компоненты вектора y_n addr_y[12 : 6] увеличивается на 1.

Фаза Upload (передача матрицы $\mathbf{R}^{-1}$ во внешнюю память, происходящая по окончании обработки выборки векторов). Передача всей матрицы занимает 8192 такта. Из памяти считывается столбец, через коммутатор MUX6 он записывается в сдвиговый регистр SHRZ, где раз в 2 такта происходит сдвиг вправо на одну компоненту. С младших 64 разрядов регистра SHRZ через коммутатор MUX7 каждый такт считывается половина элемента матрицы: в первом такте 32 разряда мнимой части, во втором такте 32 разряда действительной части. Эти действия повторяются 64 раза по числу столбцов матрицы $\mathbf{R}^{-1}$. По окончании фазы устройство ожидает нового сигнала *Start*.

Фазы Load_I и Upload выполняются один раз за время обработки выборки векторов y_n, остальные фазы повторяются 128 раз.

2. Верификация и создание прототипа

Все узлы вычислительного устройства прошли верификацию в автономном режиме на случайных направленных тестах. Одинаковые воздействия подавались на RTL-модель узла на языке *Verilog* и на функциональную модель этого узла, написанную на языке C++. Реакции обеих моделей сравнивались. Функциональная модель не описывает все подробности алгоритмов обработки чисел с плавающей запятой, используемых в узле, а опирается на операции, выполняемые микропроцессором *Pentium* 4, что упростило написание функциональной модели и ускорило ее отладку и работу.

Например, вычисление величины $AC - BD$ с 35-разрядной точностью с одним округлением после вычитания при функциональном описании узла MACR описано следующим образом.

Сначала выполняются операции умножения A и C, B и D в формате *double*.

Затем вычитание $Dif = AC - BD$ в формате *double* с округлением к нулю, при этом установлена опция *rounding precision* до 53 разрядов мантиссы, что позволяет избежать двойного округления, присущего платформе Интел. Запоминается признак *inexact*, который устанавливается в 1, если результат вычитания неточный, т. е. за разрядную сетку вытолкнута хотя бы одна 1.

Далее выполняется логическое сложение младшего разряда мантиссы *Dif* и признака *inexact* операции вычитания, это позволяет сохранить необходимую информацию для правильного округления, которое будет произведено позднее.

Потом анализируются младшие 17 разрядов мантиссы на неравенство 0. Если они отличны от нуля, то к *Dif OR inexact* прибавляется вещественное число *Round*, порядок которого на 35 меньше порядка *Dif*, скрытый бит равен 1, а дробная часть мантиссы равна 0. Так происходит округление к ближайшему. Если же младшие 17 разрядов мантиссы равны 0, то к *Dif OR inexact* число *Round* прибавляется только в случае равенства единице 35-го по старшинству разряда мантиссы (то есть, младшего остающегося в разрядной сетке разряда мантиссы).

После округления нужно обнулить младшие 18 разрядов мантиссы и преобразовать в формат, описанный в п. 1.1.

После автономной верификации узлы собирались в устройство, которое также верифицировалось на случайных направленных тестах. Устройство синтезировано на ПЛИС типа Virtex-5 xc5v1x330 [4]. Прототип устройства занимает 54 % ресурсов кристалла, рабочая частота 200 МГц. Вычисление матрицы для одной выборки занимает менее 10^{-3} с, производительность ~6,5 млрд. операций с плавающей запятой в секунду.

Заключение

В работе описана аппаратная реализация алгоритма рекурсивного вычисления комплексной матрицы 64 × 64. Проанализирована и преобразована исходная формула, разработаны функциональные модели, проведено исследование точности вычислений, предложено представление информации, облегчающее проектирование, разработаны и верифицированы узлы устройства, разработана структура специализированного вы-

числительного устройства, проведена отладка устройства, синтезирован прототип.

Производительность и точность устройства удовлетворяют требованиям.

В целях улучшения возможности различать близко расположенные цели нужно повышать точность вычислений, для этого можно:

- увеличить разрядность мантиссы используемого формата, что приводит к небольшому увеличению объема оборудования;
- уменьшить число округлений при вычислениях за счет использования метода MAF (*multiply add fused*).

Производительность устройства можно увеличить за счет:

- полной конвейеризации узлов;
- использования MAF;
- быстрого алгоритма умножения (без итераций);
- быстрого алгоритма MAC;
- более быстрого вычисления обратной величины;
- использования нескольких узлов MACR.

Реализация этих возможностей наряду с использованием заказной интегральной схемы отечественного производства с технологией 180 нм позволяет увеличить производительность в 5—10 раз по сравнению с прототипом.

Список литературы

1. **Черемисин О. П.** Адаптивные алгоритмы обработки сигналов в многоканальных приемных системах с антенными решетками // Радиотехника и электроника. 2006. Т. 51. № 9. С. 1087—1098.

2. **IEEE** Standard for Binary Floating-Point Arithmetic. ANSI/IEEE Standard No. 754. Washington, DC: American National Standards Institute, 1985.

3. **Ercegovac M., Lang T.** Digital Arithmetic. San Francisco: Morgan Kaufmann Publishers, 2004. 709 p.

4. **http:**//www.xilinx.com/support/documentation/virtex-5.htm

---

УДК 004.053

В. А. Богатырев, д-р техн. наук, проф.,
С. В. Богатырев, аспирант,
Санкт-Петербургский государственный университет информационных технологий механики и оптики,
e-mail: bva@tinuviel.ru

Надежность резервированной двухуровневой компьютерной системы при ограниченном времени обслуживания запросов

Оценена надежность двухуровневой компьютерной системы с дублированием цепей коммуникационной подсистемы, работоспособной при условии взаимосвязи между компьютерными узлами верхнего и нижнего уровней, за среднее время, не превышающее заданное пороговое значение.

Ключевые слова: *коммуникационная подсистема, отказоустойчивость, надежность, компьютерная система, среднее время пребывания.*

Введение

Высокая надежность и эффективность компьютерных систем требует структурной и временной избыточности средств обработки, хранения и передачи информации. Отказоустойчивые вычислительные системы и сети должны выдерживать, по крайней мере, однократные отказы, для этого их коммуникационные средства (коммутаторы, сетевые адаптеры (СА) и линии связи), по крайней мере, должны дублироваться [1].

Временная и структурная избыточность, как правило, взаимосвязаны. Так, структурная избыточность числа компьютерных узлов в результате распределения (распараллеливания) нагрузки приводит к сокращению времени обслуживания запросов (сокращению времени решения задачи) и, как следствие, к временной избыточности. В то же время временная избыточность расширяет возможности контроля, диагностики и управления структурно избыточной системы, а это позволяет повысить производительность системы (например, в результате оптимизаций распределения запросов).

Трудность анализа надежности отказоустойчивых компьютерных систем с дублированием связей коммуникационной подсистемы обусловлена [2—3]:

- сложным комбинаторным влиянием расположения отказавших коммуникационных средств (сетевых адаптеров, портов коммутаторов, линий связи) на надежность системы;
- пересекаемостью коммутационного оборудования, задействованного при взаимосвязи различных пар компьютеров (узлов), что вызывает необходимость выделения в каждом из дублированных коммутационных узлов оборудования, отказ которого приводит к полному отказу узла (базовое оборудование узла), и цепей,

приводящих после отказа только к снижению его коммуникационных возможностей (деградация).

Системное ограничение на допустимое время решения задачи приводит к зависимости показателя надежности системы от интенсивности входного потока запросов, при этом усложняется идентификация работоспособности состояний системы в зависимости от числа и расположения отказавших компонентов.

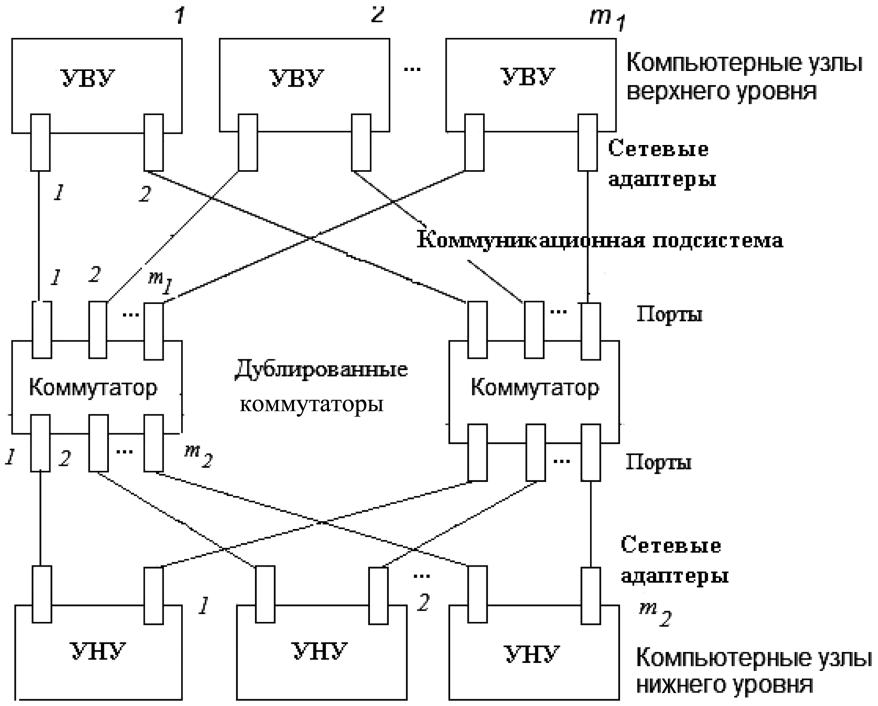


Рис. 1. Вычислительная система дублированием коммуникационных средств

Постановка задачи

Рассмотрим компьютерную систему (рис. 1), в которой выделяются узлы двух уровней — верхнего и нижнего (например, серверы и клиентские рабочие станции, или серверы и системы хранения, или компьютеры и контроллеры сбора информации и т. д.), а цепи коммуникационной подсистемы резервируются (дублируются).

Критерий работоспособности компьютерной системы заключается в выполнении запросов за ограниченное среднее время, не превышающее t_0.

Коммуникационная подсистема должна обеспечивать взаимосвязь узлов верхнего уровня (ВУ) с узлами нижнего уровня (НУ) так, чтобы общее среднее время пребывания запросов в системе (включая узлы верхнего и нижнего уровня и средства связи) не превышало t_0. Связь между узлами одного уровня не обязательна.

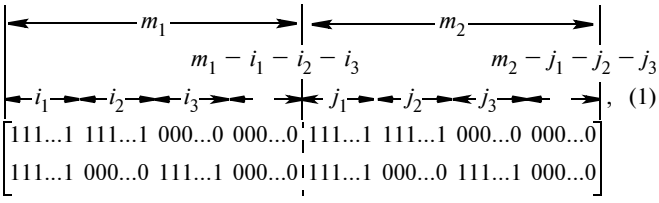
Состояние системы характеризуется числом исправных компьютерных узлов верхнего уровня (УВУ) — m_1, узлов нижнего уровня (УНУ) — m_2 и числом функционирующих коммутаторов — m_3, а также числом и расположением цепей связи компьютерных узлов (хранения или обработки данных) с коммутаторами. В исходном состоянии (до возникновения отказов) в системе имеется M_1 узлов верхнего и M_2 узлов нижнего уровня, а число коммутаторов $m_3 = 2$.

Подключение каждого из компьютерных узлов (УВУ и УНУ) к коммутатору осуществляется через цепь "Сетевой адаптер—линия—порт коммутатора" (СА—Л—ПК).

Будем считать известными данные по надежности компьютерных узлов, сетевых адаптеров, коммутаторов и линий связи. Отказы элементов системы предположим независимыми. Требуется при заданном системном ограничении на среднее время пребывания запросов оценить надежность компьютерной системы с учетом отказов цепей коммуникационной подсистемы, включая отказы дублированных коммутаторов, их портов и средств соединения (сетевых адаптеров и линий связи) с узлами ВУ и НУ.

Состояния компьютерной системы

Для исследуемых двухуровневых компьютерных систем состояние коммуникационной подсистемы при исправности m_1 из M_1, компьютерных узлов верхнего уровня, m_2 из M_2 ($m_1 + m_2 = m$) узлов нижнего уровня и работоспособности двух коммутаторов отображается матрицей $\|s_{ij}\|_{2\times m}$:



(1)

элемент которой $s_{ij} = 1$, если j-й компьютерный узел способен к взаимодействию с i-м коммутатором (исправен СА, подключающий j-й компьютерный узел через исправную линию к порту i-го исправного коммутатора), иначе $s_{ij} = 0$ [3, 4].

Состояние коммуникационной подсистемы при исправности одного коммутатора, работоспособности m_1 из M_1, компьютерных узлов верхнего уровня и m_2 из M_2 ($m_1 + m_2 = m$) узлов нижнего уровня отображается строкой

$$\underbrace{\overbrace{111\ldots1}^{i}\overbrace{000\ldots0}^{j}\overbrace{111\ldots1}^{i}\overbrace{000\ldots0}^{j}.}_{m}$$
$$\underbrace{\qquad\qquad}_{m_1}\qquad\underbrace{\qquad\qquad}_{m_2}$$

Состояния коммуникационной подсистемы в зависимости от числа отказов коммутационных элементов (число "0" в матрице $\|s_{ij}\|$) различаются как возможностью связанности УВУ и УНУ, так и средним временем пребывания запросов в системе, которое для работоспособных состояний не должно превышать заданный порог t_0.

Рассмотрим оценку вероятности работоспособности исследуемой компьютерной системы в зависимости от состояний коммутационной подсистемы, характеризуемых матрицей (1).

Вероятность работоспособного состояния компьютерной системы

Оценка вероятности P работоспособного состояния компьютерной системы основана на подсчете ее всевозможных состояний с исправностью $m_1 = 1, 2, \ldots, M_1$ узлов верхнего уровня и $m_2 = 1, 2, \ldots, M_2$ узлов нижнего уровня, отвечающих сформулированному выше условию их работоспособности (ограничения сверху среднего времени пребывания запросов в системе при связанности через коммутаторы узлов ВУ и НУ).

Надежность оценивается с учетом сформулированного критерия работоспособности системы в случае исправности двух $A_2(m_1, m_2)$ и одного $A_1(m_1, m_2)$ коммутаторов (вероятность этих событий равна p_3^2 и $2p_3(1 - p_3)$ соответственно).

Таким образом, вероятность работоспособного состояния системы

$$P = \sum_{m_1=1}^{M_1}\sum_{m_2=1}^{M_2} C_{M_1}^{m_1} C_{M_2}^{m_2} p_1^{m_1}(1-p_1)^{M_1-m_1} \times$$
$$\times\, p_2^{m_2}(1-p_2)^{M_2-m_2}[2p_3(1-p_3)A_1(m_1, m_2) +$$
$$+\, p_3^2 A_2(m_1, m_2)].$$

При оценке надежности коммуникационной подсистемы в каждом коммутаторе выделяется некоторое базовое оборудование (вероятность его работоспособности p_3), отказ которого приводит к полному отказу коммутатора, и оборудование, отнесенное к портам. Отказ оборудования, отнесенного к каждому порту, приводит к потере связанности коммутатора только с одним компьютерным узлом, причем эта связанность теряется при отказе любого элемента цепи СА—Л—ПК. Вероятность сохранения связанности компьютерного узла с одним коммутатором

$$p_a = p_{ca} p_l p_s,$$

где p_{ca}, p_l, p_s — вероятности исправности элементов цепи СА—Л—ПК.

Вероятность работоспособности коммутационной подсистемы при исправности двух $A_2(m_1, m_2)$ и одного $A_1(m_1, m_2)$ коммутатора вычисляется по формулам

$$A_1(m_1, m_2) = \sum_{i=1}^{m_1}\sum_{j=1}^{m_2} \delta_1(i, j)\, C_{m_1}^{i} C_{m_2}^{i} \times$$
$$\times\, p_a^i (1-p_a)^{m_1-i} p_a^{m_1}(1-p_a)^{m_2-j};$$
$$A_2(m_1, m_2) =$$
$$= \sum_{i_1=0}^{m_1}\sum_{i_2=0}^{m_1-i_1}\sum_{i_3=0}^{m_1-i_1-i_2} C_{m_1}^{i_1} C_{m_1-i_1}^{i_2} C_{m_1-i_i-i_2}^{i_3} \times$$
$$\times\, p_a^{2i_1+i_2+i_3}(1-p_a)^{2m_1-2i_1-i_2-i_3} \times$$
$$\times \sum_{j_1=0}^{m_2}\sum_{j_2=0}^{m_2-j_1}\sum_{j_3=0}^{m_2-j_1-j_2} \delta_2(i_1, i_2, i_3, j_1, j_2, j_3)\, C_{m_2}^{j_1} C_{m_2-j_1}^{j_2} \times$$
$$\times\, C_{m_2-j_1-j_2}^{j_3} p_a^{2j_1+j_2+j_3}(1-p_a)^{2m_2-2j_1-j_2-j_3},$$

где $\delta_1(i, j)$ и $\delta_2(i_1, i_2, i_3, j_1, j_2, j_3)$ — условия (критерий) работоспособности состояния коммуникационной подсистемы при исправности одного и двух коммутаторов.

В системе имеются ограничения на задержку обслуживания запросов, поэтому вычисление надежности требует оценки среднего времени пребывания запросов в зависимости от числа и расположения отказавших телекоммуникационных средств.

Среднее время пребывания запросов при исправности одного коммутатора

При представлении каждого узла системой массового обслуживания M/M/1 и равномерном распределении запросов между всеми узлами среднее время пребывания запросов в системе (при прохождении каждого запроса последовательно через один УВУ, коммутатор и УНУ) определяется формулой

$$T_0(i, j) = \frac{\nu_1}{1 - \frac{\Lambda\nu_1}{i}} + \frac{\nu_3}{1-\Lambda\nu_3} + \frac{\nu_2}{1-\frac{\Lambda\nu_2}{j}},$$

где Λ — интенсивность поступающего в систему потока запросов; ν_1, ν_2, ν_3 — средние времена выполнения запроса в узлах ВУ, НУ и в коммуникационной подсистеме.

Определив для каждого состояния среднее время пребывания запросов в системе, его работоспособность найдем в соответствии с условием

$\delta_1(i, j) = 1$, если $T_0(i, j) < t_0$, иначе $\delta_1(i, j) = 0$.

Среднее время пребывания запросов в системе при исправности двух коммутаторов

Среднее время пребывания запросов в системе определяется как

$$T(i_1, i_2, i_3, j_1, j_2, j_3) = T_1(i_1, i_2, i_3, j_1, j_2, j_3) + \\ + T_2(i_1, i_2, i_3, j_1, j_2, j_3) + T_3(i_1, i_2, i_3, j_1, j_2, j_3),$$

где $T_1(i_1, i_2, i_3, j_1, j_2, j_3)$, $T_2(i_1, i_2, i_3, j_1, j_2, j_3)$, $T_3(i_1, i_2, i_3, j_1, j_2, j_3)$ — средние времена обслуживания в узлах верхнего, нижнего уровней и в коммутаторах.

Работоспособность состояния определяется условием не превышения среднего времени пребывания запросов в системе заданного порога t_0, т. е.

$$\delta_2(i_1, i_2, i_3, j_1, j_2, j_3) = 1, \\ \text{если } T_1(i_1, i_2, i_3, j_1, j_2, j_3) < t_0, \\ \text{иначе } \delta_2(i_1, i_2, i_3, j_1, j_2, j_3) = 0.$$

Среднее время пребывания запросов в узлах верхнего уровня. При определенных состояниях системы некоторые исправные узлы ВУ могут оказаться не способными к приему запросов на обслуживание вследствие того, что коммутационная подсистема не обеспечивает их связанность с узлами НУ. Такие состояния возможны даже при подключении исправных узлов к исправным коммутаторам через исправные цепи СА—Л—ПК. Например, для состояния коммутационной подсистемы, представляемого матрицей $\|s_{ij}\|_{2\times m}$ вида

$$\begin{bmatrix} \overbrace{11111...1000}^{i_2} & 00000 \\ 11100...0111 & 11111 \end{bmatrix},$$

i_2 узлов ВУ, подключенных только к первому коммутатору (верхняя строка матрицы), не доступны для обслуживания входного потока запросов, так как коммуникационная подсистема не может обеспечить их связанность с узлами НУ (хотя все узлы верхнего и нижнего уровней подключены к не отказавшим коммутаторам через исправные цепи СА—Л—ПК).

Коммуникационная подсистема, состояние которой отображается матрицей (1), обеспечивает работоспособность i_1 узлов ВУ, связанных (через исправные цепи) с двумя коммутаторами, если имеются узлы НУ, подключенные хотя бы к одному из двух коммутаторов, так как в этом случае имеется возможность передать запросы на продолжение обслуживания в узлы НУ.

Таким образом, число способных принять на обслуживание запросы узлов верхнего уровня, подключенных к двум коммутаторам, равно

$$I_1(i_1, j_1, j_2, j_3) = i_1\beta_1(i_1, j_1, j_2, j_3),$$

причем $\beta_1(i_1, j_1, j_2, j_3) = 1$, если $(j_1 + j_2 + j_3) > 1$, иначе $\beta_1(i_1, j_1, j_2, j_3) = 0$.

Для $i_2(i_3)$ узлов ВУ, подсоединенных только к первому (второму) коммутатору, возможен прием запросов на обслуживание, если соответствующий коммутатор связан с исправными узлами НУ, т. е. если $j_1 + j_2 > 0$ ($j_1 + j_3 > 0$).

Таким образом, число узлов ВУ, подключенных только к первому коммутатору, которые способны принять на обслуживание запросы, равно:

$$I_2(i_2, j_1, j_2, j_3) = i_2\beta_2(i_2, j_1, j_2, j_3),$$

где $\beta_2(i_2, j_1, j_2, j_3) = 1$, если $(j_1 + j_2) > 1$, иначе $\beta_2(i_2, j_1, j_2, j_3) = 0$,

а ко второму коммутатору —

$$I_3(i_3, j_1, j_2, j_3) = i_3\beta_2(i_1, j_1, j_2, j_3),$$

где $\beta_3(i_3, j_1, j_2, j_3) = 1$, если $(j_1 + j_3) > 1$, иначе $\beta_3(i_3, j_1, j_2, j_3) = 0$.

Общее число узлов ВУ, которые способны принять на обслуживание запросы, равно

$$I(i_1, i_2, i_3, j_1, j_2, j_3) = \\ = I_1(i_1, j_1, j_2, j_3) + I_2(i_2, j_1, j_2, j_3) + I_3(i_3, j_1, j_2, j_3).$$

Среднее время пребывания запросов в узлах ВУ (при их равномерной загрузке) вычисляется по формуле

$$T_1(i_1, i_2, i_3, j_1, j_2, j_3) = \frac{\nu_1}{1 - \dfrac{\Lambda\nu_1}{I(i_1, i_2, i_3, j_1, j_2, j_3)}}.$$

Среднее время пребывания запросов в коммуникационной подсистеме. Среднее время пребывания запросов в коммуникационной подсистеме при исправности двух коммутаторов определяется формулой

$$T_3(i_1, i_2, i_3, j_1, j_2, j_3) = \pi_1 T_{31}(i_1, i_2, i_3, j_1, j_2, j_3) + \\ + \pi_2 T_{32}(i_1, i_2, i_3, j_1, j_2, j_3),$$

где π_1, π_2 — доля потока пакетов (запросов), передаваемых от УВУ к УНУ соответственно через первый и второй коммутатор:

$$\pi_1 = \frac{0{,}5I_1(i_1, i_2, i_3, j_1, j_2, j_3) + I_2(i_1, i_2, i_3, j_1, j_2, j_3)}{I(i_1, i_2, i_3, j_1, j_2, j_3)};$$

$$\pi_2 = \frac{0{,}5I_1(i_1, i_2, i_3, j_1, j_2, j_3) + I_3(i_1, i_2, i_3, j_1, j_2, j_3)}{I(i_1, i_2, i_3, j_1, j_2, j_3)};$$

$T_{31}(i_1, i_2, i_3, j_1, j_2, j_3)$, $T_{32}(i_1, i_2, i_3, j_1, j_2, j_3)$ — среднее время задержки (пребывания) запросов в первом и втором коммутаторе.

Если коммутаторы одинаковы, то

$$T_{31}(i_1, i_2, i_3, j_1, j_2, j_3) = \frac{\nu_3}{1 - \pi_1 \nu_3 \Lambda};$$

$$T_{32}(i_1, i_2, i_3, j_1, j_2, j_3) = \frac{\nu_3}{1 - \pi_2 \nu_3 \Lambda}.$$

Среднее время пребывания запросов в узлах нижнего уровня. Среднее время пребывания запросов в узлах НУ зависит от входного потока запросов и от числа узлов НУ, для которых коммуникационная подсистема обеспечивает связанности с узлами ВУ.

При определенных состояниях коммуникационной подсистемы некоторые работоспособные узлы НУ могут оказаться не доступными для распределения запросов даже при их подключении к исправному коммутатору через исправные цепи СА—Л—ПК.

Например, для состояния коммутационной подсистемы, отображаемого матрицей $\|s_{ij}\|_{2\times m}$ вида

$$\begin{bmatrix} 0000 & \vdots & \overbrace{1111\ldots}^{j_2}1000 \\ 1111 & \vdots & 11100\ldots0111 \end{bmatrix},$$

j_2 узлов НУ (правая подматрица), подключенных только к первому коммутатору (верхняя строка матрицы), не доступны для обслуживания запросов, передаваемых через коммуникационную подсистему от узлов ВУ (левая подматрица).

Коммуникационная подсистема, состояние которой отображается матрицей (1), обеспечивает работоспособность j_1 узлов НУ, связанных (через исправные цепи СА—Л—ПК) с двумя коммутаторами, если имеются узлы ВУ, подключенные хотя бы к одному из двух коммутаторов. Таким образом, число узлов НУ, подключенных к двум коммутаторам, которые способны принять на обслуживание запросы, равно

$$J_1(j_1, i_1, j_2, i_3) = j_1 \psi_1(j_1, i_1, i_2, i_3),$$

причем $\psi_1(j_1, i_1, i_2, i_3) = 1$, если $(i_1 + i_2 + i_3) > 1$, иначе $\psi_1(j_1, i_1, i_2, i_3) = 0$.

Для $j_2(j_3)$ число узлов НУ, подсоединенных только к первому (второму) коммутатору, возможно обслуживание запросов, если этот коммутатор связан через исправные цепи СА—Л—ПК с работоспособными узлами ВУ, т. е. когда $i_1 + i_2 > 0$.

Таким образом, число узлов НУ, подключенных только к первому коммутатору, которые способны принять на обслуживание запросы, равно

$$J_2(j_2, i_1, i_2, i_3) = j_2 \psi_2(j_2, i_1, i_2, i_3),$$

где $\psi_2(j_2, i_1, i_2, i_3) = 1$, если $(i_1 + i_2) > 1$, иначе $\psi_2(j_2, i_1, i_2, i_3) = 0$;
аналогично число узлов НУ, подключенных только ко второму коммутатору, которые способны принять на обслуживание запросы, равно

$$J_3(j_3, i_1, i_2, i_3) = j_3 \psi_3(j_3, i_1, i_2, i_3),$$

причем $\psi_2(j_2, i_1, i_2, i_3) = 1$, если $(i_1 + i_3) > 1$, иначе $\psi_2(j_3, i_1, i_2, i_3) = 0$.

Общее число узлов НУ, которые доступны через коммуникационную подсеть для обслуживания запросов от узлов ВУ, равно

$$j(i_1, i_2, i_3, j_1, j_2, j_3) = \\ = J_1(j_1, i_1, i_2, i_3) + J_2(j_2, i_1, i_2, i_3) + J_3(j_3, i_1, i_2, i_3).$$

Среднее время пребывания запросов в узлах НУ при равномерной загрузке всех доступных через коммуникационную подсистему узлов НУ вычисляется по формуле

$$T_2(i_1, i_2, i_3, j_1, j_2, j_3) = \frac{\nu_2}{1 - \dfrac{\Lambda \nu_2}{J(i_1, i_2, i_3, j_1, j_2, j_3)}}.$$

Предельная интенсивность запросов, обслуживаемых компьютерной системой

При исправности двух коммутаторов предельно допустимая интенсивность запросов Λ_0, не вызывающая перегрузки узлов системы, находится из условия

$$\Big[\Big(I(i_1, i_2, i_3, j_1, j_2, j_3) > 0\Big) \wedge \\ \wedge \Big(\frac{\Lambda_0 \nu_1}{I(i_1, i_2, i_3, j_1, j_2, j_3)} < 1\Big)\Big] \wedge \\ \wedge \Big[J(i_1, i_2, i_3, j_1, j_2, j_3 > 0) \wedge \\ \wedge \Big(\frac{\Lambda_0 \nu_2}{J(i_1, i_2, i_3, j_1, j_2, j_3)} < 1\Big)\Big] \wedge (\pi_1 \nu_3 \Lambda_0 < 1) \wedge \\ \wedge (\pi_2 \nu_3 \Lambda_0 < 1).$$

При исправности одного коммутатора предельно допустимая интенсивность запросов, не вызывающая перегрузки в узлах (включая коммутатор) системы, находится из условия

$$\Big(\frac{\Lambda_0 \nu_1}{i} < 1\Big) \wedge \Big(\frac{\Lambda_0 \nu_2}{j} < 1\Big) \wedge (\Lambda_0 \nu_3 < 1).$$

Если предельно допустимая интенсивность запросов, рассчитываемая при исправности одного коммутатора, не удовлетворяет условиям функционирования системы, то соответствующие со-

стояния исключаются из множества работоспособных состояний.

Представленные в статье зависимости позволяют определить предельную интенсивность входного потока запросов при условии не превышения среднего времени пребывания запросов в системе, а также не нарушения заданной в техническом задании нижней грани допустимой надежности системы.

Пример расчета надежности компьютерной системы

Оценку надежности системы проведем в предположении экспоненциального распределения времени между отказами элементов. Вероятность сохранения работоспособности узлов верхнего и нижнего уровня, цепи СА—Л—ПК и базового оборудования коммутатора в течение времени t: $p_1 = \exp(-\lambda_1 t)$, $p_2 = \exp(-\lambda_2 t)$, $p_3 = \exp(-\lambda_3 t)$, $p_a = \exp(-\lambda_a t)$, где λ_1, λ_2, λ_3, λ_a — суммарные интенсивности отказов соответственно УВУ, УНУ базового оборудования коммутатора и цепи СА—Л—ПК (рассматриваемая зависимость вероятности исправности компонентов от времени приводит к зависимости $P(t)$ надежности системы от времени). Будем считать: $\lambda_1 = \lambda_2 = \lambda$, $\lambda_3 = \alpha\lambda$, $\lambda_a = \lambda_{ca} + \lambda_{pk} + \lambda_l$, а λ_{ca}, λ_l, λ_{pk} — интенсивности отказов компонентов цепи СА—Л—ПК, причем предположим, что $\lambda_a = \alpha_1\lambda$.

В исходном состоянии (до отказов) число компьютерных узлов верхнего и нижнего уровня равно $M_1 = 5$, $M_2 = 6$. При расчете будем считать, что $\lambda = 0{,}0002$ (1/ч), $\alpha = 0{,}5$ $\nu_1 = 1$, $\nu_3 = 0{,}1$, $\nu_3 = 1{,}5$ (с).

При заданном ограничении на среднее время пребывания запросов в системе $t_0 = 100$ с зависимость $P(t)$ вероятности работоспособности компьютерной системы от времени работы t (ч) представлена на рис. 2 кривыми *1*, *2* для интенсивности входного потока запросов $\Lambda = 2{,}99$ (1/ч) и $\Lambda = 1{,}99$ (1/ч) соответственно. Кривая *3* дает верхнюю оценку надежности $P_h(t)$ при отсутствии ограничений на время пребывания запросов в системе. Рис. 2, *а* представляет оценку надежности при времени работы системы до 700 ч, а рис. 2, *б* — до 250 ч.

На рис. 3 представлена зависимость разницы оценки системы от времени t с учетом ограничений на среднее время пребывания запросов в системе и без их учета $\Delta = P_h(t) - P(t)$. Рис. 3, *а* представляет диапазон времени t работы системы до 700 ч, а рис. 3, *б* — до 250 ч. Кривая *1* соответствует разнице вероятности $\Delta = P_h(t) - P(t)$ при $\Lambda = 2{,}99$ ч$^{-1}$, а кривая *2* — при $\Lambda = 1{,}99$ ч$^{-1}$.

Приведенные графики подтверждают необходимость учета влияния на надежность системы ог-

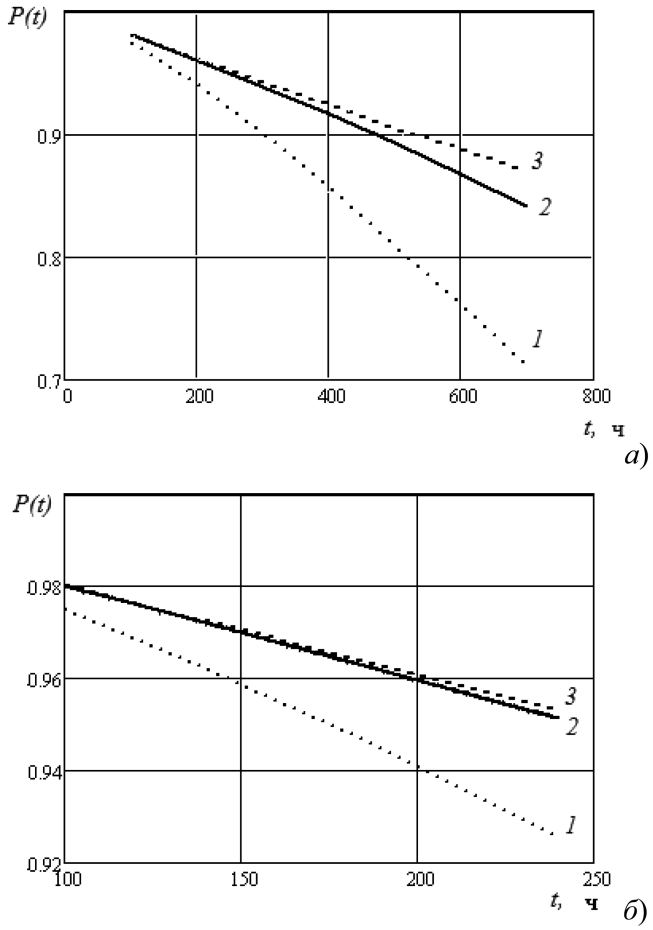


Рис. 2. Вероятность работоспособности системы от времени работы *t*, ч

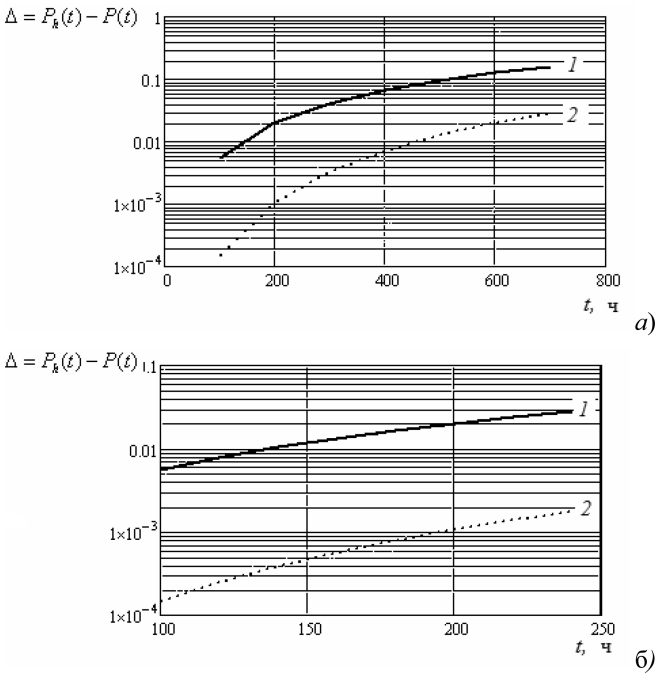


Рис. 3. Разница оценки надежности системы с учетом и без учета ограничений на среднее время пребывания запросов в системе

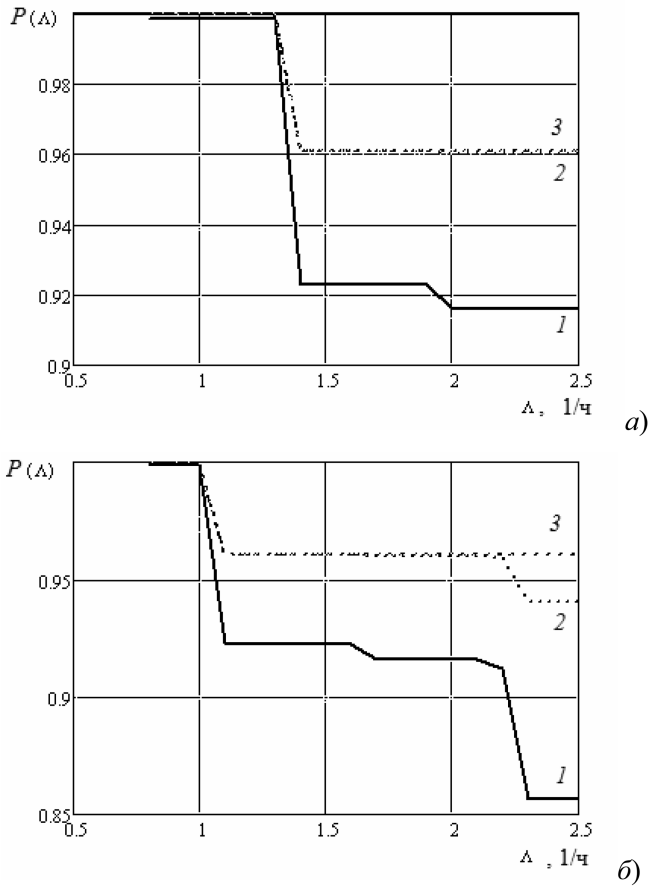


Рис. 4. Зависимость вероятности работоспособности системы от интенсивности входного потока Λ

раничений на среднее время пребывания запросов в системе. При грубых расчетах для систем, не критичных к времени обработки запросов, ограничениями на среднее время пребывания запросов можно пренебречь.

Уровень надежности исследуемых систем (вероятности работоспособного состояния) и погрешность $\Delta = P_h - P$, возникающая из-за неучета ограничений на допустимое время пребывания запросов t_0, зависят от интенсивности входного потока Λ и от порога t_0.

Зависимость надежности компьютерной системы от интенсивности входного потока Λ (1/ч) представлена на рис. 4 кривыми *1*, *2* соответственно для наработки $t = 200$ ч и $t = 400$ ч. Кривая *3* отражает верхнюю оценку надежности при отсутствии (неучете) ограничений на время пребывания запросов в системе. Рис. 4, *а* представляет расчеты при ограничении на среднее время пребывания запросов в системе $t_0 = 100$ ч, а рис. 4, *б* — при $t_0 = 10$ ч.

На рис. 5 представлена зависимость разницы оценки надежности системы с учетом ограничений на среднее время пребывания запросов в системе и без их учета $\Delta = P_h(t) - P(t)$ от интенсивности входного потока Λ (1/ч). Кривая *1* представляет разницу вероятности $\Delta = P_h(t) - P(t)$ при времени работы системы (наработке) $t = 200$ ч, а кривая *2* — при времени $t = 400$ ч. Рис. 5, *а* соответствует ограничению на среднее время пребывания запросов в системе $t_0 = 100$ ч, а рис. 5, *б* — $t_0 = 10$ ч.

Графики на рис. 4 и 5 позволяют оценивать влияние на надежность значения верхнего допустимого порога пребывания запросов в системе, а также судить о погрешности оценки надежности системы, если указанное ограничение не учитывать.

Таким образом, предложена оценка надежности компьютерной системы с дублированием коммуникационных узлов и их связей при требовании взаимосвязи компьютерных узлов верхнего уровня с узлами нижнего уровня через коммуникационную подсистему за среднее время, не превышающее заданное предельное значение.

Оценка надежности учитывает число и расположение отказов цепей СА—Л—ПК компьютерных узлов с коммутаторами и влияние расположения этих отказов на среднее время пребывания запросов в компьютерной системе.

При оценке надежности в каждом коммутаторе выделяются некоторое базовое оборудование, от-

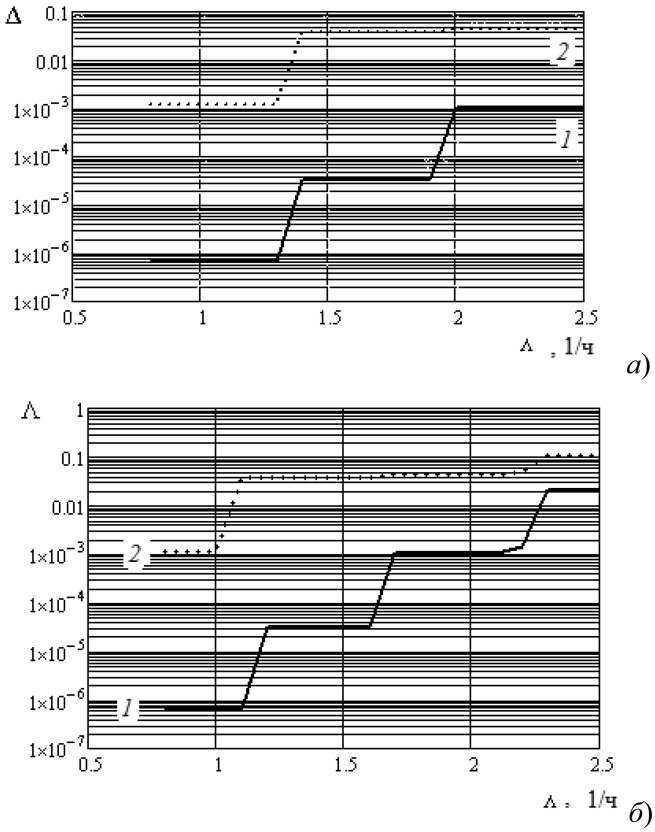


Рис. 5. Разница оценки надежности системы с учетом и без учета ограничений на среднее время пребывания запросов в системе

каз которого приводит к полному отказу коммутатора, и оборудование каждого порта, отказ которого приводит к потере связанности только с одним компьютерным узлом.

Предложенные модели надежности могут быть применены для анализа двухуровневых компьютерных систем и сетей, предусматривающих взаимосвязь между узлами верхнего и нижнего уровней (локальных сетей с архитектурой "клиент—сервер", сетей хранения данных, систем связи промышленных компьютеров с датчиками и исполнительными механизмами и т. п.).

Полученные результаты могут использоваться при обосновании (выборе) структуры отказоустойчивых компьютерных систем с учетом структурной составляющей влияния отказов (потеря связанности компьютерных узлов вследствие отказов цепей их связи) и влияния отказов на превышение допустимого времени обработки запросов в системе.

Список литературы

1. **Олифер В. Г., Олифер Н. А.** Компьютерные сети. Принципы, технологии. Протоколы, СПб.: Питер. 2006.
2. **Богатырев В. А.** Отказоустойчивость и сохранение эффективности функционирования многомагистральных распределенных вычислительных систем // Информационные технологии. 1999. № 9. С. 44—48.
3. **Богатырев В. А.** Комбинаторно-вероятностная оценка надежности и отказоустойчивости кластерных систем / Приборы и системы. Управление, контроль, диагностика. 2006. № 6. С. 21—26.
4. **Богатырев В. А.** Надежность и эффективность резервированных компьютерных сетей // Информационные технологии. 2006. № 9. С. 26—30.
5. **Богатырев В. А.** Комбинаторный метод оценки отказоустойчивости многомагистрального канала // Методы менеджмента качества. 2000. № 4. С. 30—35.

СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ

УДК 621.3.049.771.14

А. Л. Стемпковский, акад. РАН, директор,
А. Л. Глебов, д-р техн. наук, зав. сектором,
С. В. Гаврилов, д-р техн. наук, зав. сектором, **О. Н. Гудкова,** аспирант,
Институт проблем проектирования в микроэлектронике РАН, г. Москва,
e-mail: Gudkova_O@ippm.ru

Вероятности напряженного состояния транзисторов для временного анализа с учетом электротемпературной нестабильности

Нестабильность, вызванная отрицательным смещением и температурой (НОСТ, англоязычная аббревиатура — NBTI) становится одним из основных механизмов, вызывающих деградацию быстродействия интегральных схем. Хорошо известно, что НОСТ влияет на p-канальные МОП-транзисторы в ходе работы схемы, и деградация имеет место, когда p-транзистор находится в проводящем состоянии. Поэтому точный анализ НОСТ-деградации требует рассмотрения логики работы схемы. Деградация конкретного p-транзистора зависит от той части времени работы схемы, в течение которой транзистор находится в напряженном состоянии, иначе говоря, от вероятности напряженного состояния. В настоящей работе предлагается алгоритм корректного вычисления вероятности напряженного состояния для каждого p-транзистора в сложном КМОП-вентиле. По сравнению с простым "наивным " подходом предлагаемый алгоритм учитывает два дополнительных фактора: корреляции между сигналами на входах вентиля и потенциал питания, поступающий через "нижний" узел p-транзистора. Численные эксперименты подтверждают важность учета обоих этих факторов.

Ключевые слова: *временной анализ СБИС, КМОП-вентиль, нестабильность, вызванная отрицательным смещением и температурой (НОСТ).*

Введение

Старение интегральных схем становится важным фактором, ограничивающим их быстродействие, для современных (глубоко субмикрометровых и нанометровых) технологий. Нестабильность, вызванная отрицательным смещением и температурой (НОСТ, мы предлагаем такую аббревиатуру как перевод англоязычной аббревиатуры NBTI — Negative Bias Temperature Instability) является одним из основных механизмов деградации схем. НОСТ влияет в основном на *p*-канальные МОП-транзисторы и приводит к деградации порогового напряжения (V_{th}). НОСТ-деградация происходит в течение всего времени, когда на затвор *p*-транзистора подается отрицательное смещение (напряжение). В общем случае НОСТ мо-

жет приводить к увеличению задержки схемы более чем на 20 %, а в некоторых исключительных случаях даже к нарушению логики функционирования схемы. НОСТ возникает либо при отрицательном смещении затвора транзистора, либо при повышении его температуры, но наиболее сильный эффект наблюдается при сочетании обоих факторов. Точная оценка деградации быстродействия требует учета не только влияния напряжения питания (V_{dd}) и температуры (T), но также влияния логики переключательной активности схемы.

В ряде работ (см., например, [1—5]) было исследовано влияние НОСТ на деградацию быстродействия. Позднее, в работе [6] была разработана достаточно точная модель деградации на транзисторном уровне, учитывающая время напряженного состояния транзистора, коэффициент заполнения тактового сигнала, а также условия эксплуатации интегральной схемы. Моделирование конкретной последовательности входных векторов для оценки деградации схемы является, конечно, невыполнимой задачей. Поэтому при анализе НОСТ-деградации мы вынуждены рассматривать вероятностную модель сигнала, т. е. вычислять вероятности значения сигнала в каждом узле схемы и коэффициенты корреляции между сигналами.

В работе [7] указанная модель упрощается для непосредственного предсказания деградации задержки в предположении линейной и единственной деградации V_{th}. Очевидно, однако, что вентили в схеме (библиотеке) могут иметь более одного p-транзистора. Каждый из этих p-транзисторов находится в своем напряженном состоянии и имеет свою, отличную от других, деградацию V_{th}. Кроме этого, деградация V_{th} каждого p-транзистора зависит от вероятности значения соответствующего входного сигнала, а также от его корреляций с состояниями других p-транзисторов, с которыми рассматриваемый транзистор соединен последовательно. В [7] показано, что для каждого критического пути существует своя вероятность значения сигнала, приводящая к максимальной (т. е. наихудшей) деградации задержки. Подход, предлагаемый в [7], предполагает, что корреляции между сигналами отсутствуют. Однако это предположение может приводить как к излишне пессимистическим, так и к излишне оптимистическим предсказаниям задержки схемы. Рассмотрим, например, такую же схему как и в [7] (показана на рис. 1, *а*). В этом случае нетрудно увидеть, что трехвходовый вентиль NOR (t_{p4}) имеет три последовательно соединенных p-транзистора. Переключающийся p-транзистор имеет вероятность значения логического нуля на затворах (1-α), тогда как другие два последовательных p-транзистора являются проводящими и находятся в напряженном состоянии с вероятностью 1. Рассмотрим теперь случай, когда значения сигналов на боковых входах пути имеют различные вероятности. Например, если все эти вероятности равны 0,5 (как показано на рис. 1, *б*), то легко показать, что такая ситуация приводит к большей деградации задержки по сравнению со схемой на рис. 1, *а*. Таким образом, для более точного предсказания НОСТ-деградации необходимо идентифицировать вероятности значений сигналов и корреляции между сигналами, существующие в схеме.

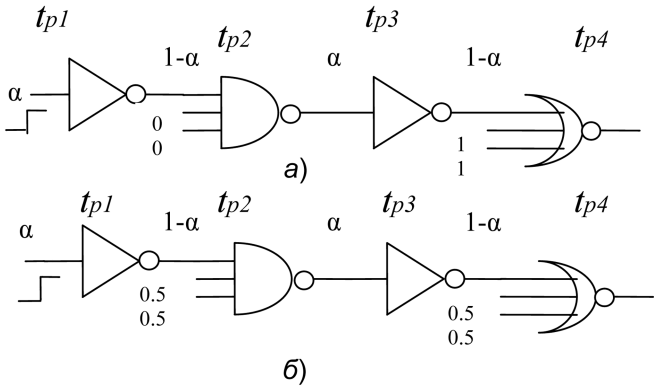


Рис. 1. Вероятности значения сигнала на критическом пути

В данной работе предложен оригинальный алгоритм корректного вычисления вероятности напряженного состояния (*Pstress*) для каждого p-транзистора в схеме. Значения вероятностей логического нуля (*PLZ*) рассчитываются для всех узлов схемы, начиная от первичных и далее последовательно до выходов. В той же последовательности рассчитываются также парные корреляции между сигналами. После этого для каждого p-транзистора в КМОП-вентиле вычисляется *Pstress*.

В данной работе используется следующая терминология. Пусть p-транзистор x соединен с узлами M и N (через контакты истока и стока). Если существует несамопересекающийся путь в последовательно-параллельной (ПП) верхней цепи, идущий от узла питания к выходу вентиля и проходящий через транзистор x, причем M на пути предшествует N, то будем говорить, что M — верхний узел, а N — нижний узел транзистора x. Простейший алгоритм нахождения *Pstress* для x рассматривает все пути от узла питания до верхнего узла транзистора x. В противоположность этому наивному подходу предлагаемый нами алгоритм дополнительно учитывает следующие два фактора: корреляции между сигналами на входах вентиля и потенциал питания, приходящий через нижние узлы p-транзисторов. Результаты численных экспериментов показывают важность учета обоих этих факторов.

Оставшаяся часть этой работы имеет следующую структуру: раздел 1 описывает вычисление вероятностей значений сигнала и корреляций в узлах схемы. Раздел 2 содержит описание предлагаемого алгоритма вычисления *Pstress* для каждого *p*-транзистора в стандартном КМОП-вентиле. Раздел 3 представляет экспериментальные результаты.

1. Расчет вероятностей значений сигнала и корреляций

НОСТ влияет только на *p*-транзисторы, находящиеся в проводящем состоянии. Для этого контакт затвора *p*-транзистора должен находиться в логическом "0". Учет вероятности того, что значение сигнала — логический "0" и его распространение являются необходимыми условиями для правильного вычисления деградации задержки с учетом НОСТ-эффектов. В данном разделе дается краткое описание метода вычисления вероятности значения сигнала для каждого узла схемы с учетом корреляций между сигналами. Мы предполагаем, что вероятности для первичных входов заданы. Для вероятности логического "0" используем обозначение *PLZ*.

Корреляции между сигналами. Мы предполагаем, что каждый сигнал a в комбинационной схеме (либо первичный вход, либо выход любого вентиля) характеризуется величиной $PLZp_a$. Для описания корреляции между двумя сигналами a и b введем коэффициент корреляции сигналов (SC) по аналогии с [8, 9]:

$$SC_{ij}^{ab} = \frac{p(a = i \& b = j)}{p(a = i)p(b = j)}, \qquad (1)$$

где $i, j = 0; 1$.

Мы используем основное предположение о том, что только парные корреляции существенны. Иначе говоря, мы пренебрегаем корреляцией любых двух сигналов к третьему и т. д. В этом случае вероятность сложного события может быть приближенно вычислена по формуле

$$p\Big(\prod_{k=1}^{n}(a_k = i_k)\Big) =$$

$$= \prod_{k=1}^{n} p(a_k = i_k) \prod_{1 \le k \le l \le n} SC_{i_k i_l}^{a_k a_l}. \qquad (2)$$

Поскольку

$$p(a = i \& b = j) = p(a = i/b = j)p(b = j) = \\ = p(b = j/a = i)p(a = i), \qquad (3)$$

где $p(X/Y)$ обозначает вероятность события X при условии Y, то

$$SC_{ij}^{ab} = \frac{p(a = i/b = j)}{p(a = i)} = \frac{p(b = j/a = i)}{p(b = j)}. \qquad (4)$$

Следовательно, мы имеем следующие соотношения для четырех коэффициентов SC^{ab}:

$$\sum_{i=0,1} SC_{ij}^{ab} p(a = i) = 1, j = 0; 1; \\ \sum_{i=0,1} SC_{ij}^{ab} p(b = j) = 1, i = 0; 1. \qquad (5)$$

Матрица системы (5) имеет ранг 3 [8], поэтому, если мы знаем SC_{00}^{ab}, мы можем вычислить три других коэффициента по формулам:

$$SC_{01}^{ab} = \frac{1 - SC_{00}^{ab} p_b}{1 - p_b}; \qquad (6)$$

$$SC_{10}^{ab} = \frac{1 - SC_{00}^{ab} p_a}{1 - p_a}; \qquad (7)$$

$$SC_{11}^{ab} = \frac{1 - SC_{10}^{ab} p_b}{1 - p_b}. \qquad (8)$$

Расчет вероятности значений сигнала в узлах схемы. Для вычисления *PLZ* для выхода вентиля должны быть известны вероятности для всех входов вентиля и коэффициенты SC для всех пар входов вентиля. Все возможные входные векторы вентиля могут быть разделены на два множества:

- V_0 — множество входных векторов вентиля, для которых выход вентиля равен 0;
- V_1 — множество входных векторов вентиля, для которых выход вентиля равен 1.

Для сложного события $y = 0$ можно записать:

$$(y = 0) = \sum_{I \in V_0} \prod_{k=1}^{n} (x_k = i_k), \qquad (9)$$

где n — число входов вентиля; x_k — сигналы на входах вентиля (переменные); $I = (i_1, \ldots, i_n)$ — входной вектор вентиля. Векторы под дизъюнкцией в (9) являются взаимно исключающими, поэтому

$$p(y = 0) = \sum_{I \in V_0} p\Big(\prod_{k=1}^{n} (x_k = i_k)\Big). \qquad (10)$$

Используя (2) и пренебрегая корреляциями более высоких порядков, получаем следующую формулу для расчета вероятности логического нуля на выходе вентиля:

$$p(y = 0) = \sum_{I \in V_0} \prod_{k=1}^{n} (p(x_k = i_k) \prod_{k < l \le n} SC_{i_k i_l}^{x_k x_l}). \qquad (11)$$

Отметим, что описанные вычисления можно сделать более эффективными, используя BDD-представление для КМОП-вентилей (см. ниже).

Распространение корреляций между сигналами. Для корректной обработки каждого вентиля мы также должны распространять по схеме коэффициенты корреляции *SC*. Рассмотрим тот же вентиль, что и в начале предыдущего подраздела. Пусть z — любой сигнал с известной вероятностью и известными *SC* с каждым входом вентиля. Необходимо решить задачу распространения *SC* сигнала z с входов вентиля на его выход. В соответствии с определением (1)

$$SC_{00}^{zy} = \frac{p(z = 0 \& y = 0)}{p(z = 0)p(y = 0)}. \qquad (12)$$

Рассматривая $(z = 0 \& y = 0)$ как сложное событие (аналогично рассмотрению в предыдущем подразделе) и вычисляя его вероятность, получаем следующую формулу для распространения *SC* через вентиль:

$$SC_{00}^{zy} =$$

$$= \frac{\sum\limits_{I \in V_0} \prod\limits_{k=1}^{n} \left(p(x_k = i_k) SC_{0 i_k}^{z x_k} \prod\limits_{k < l \le n} SC_{i_k i_l}^{x_k x_l} \right)}{p(y = 0)}. \qquad (13)$$

Как и в предыдущем подразделе, эта формула может быть также записана с использованием BDD-представления КМОП-вентиля. Остальные три коэффициента *SC* могут быть вычислены по формулам (6)—(8).

2. Вероятности напряженного состояния для КМОП вентиля

Допустим, что для каждого входа вентиля и для каждой пары входов вентиля найдены соответственно вероятности логического нуля (*PLZ*) и парные корреляции. После этого может быть вычислена вероятность напряженного состояния (*Pstress*) для каждого *p*-транзистора в вентиле.

Предлагаемая в данной работе версия алгоритма основана на SP-BDD-представлении КМОП-вентиля. SP-BDD (*series-parallel* BDD, или последовательно-параллельная диаграмма двоичных решений) — это BDD специального вида, крайне удобная для представления КМОП-вентиля в различных целях. Она была первоначально предложена в [10] и использовалась, например, для быстрого вычисления потребляемой мощности [11] и логического ресинтеза [12] КМОП-схем.

SP-BDD является каноническим представлением КМОП-вентиля. Она полностью описывает как Булеву функцию вентиля, так и его детальную структуру на уровне транзисторов. Каждая вершина SP-BDD соответствует паре МОП-транзисторов: один из верхней и один из нижней цепи вентиля. Рис. 2 иллюстрирует SP-BDD-представление КМОП-вентиля.

Предварительно нами был опробован простейший алгоритм нахождения *Pstress* на основе SP-BDD. Этот алгоритм учитывает все пути от узла питания к верхнему узлу *p*-транзистора x (не включающие транзистор x). Точность этого "наивного" алгоритма может быть значительно улучшена. Однако для того, чтобы выполнить это улучшение, необходимо решить две проблемы.

Первая проблема состоит в том, что мы должны учесть все парные корреляции между сигналами на входах вентиля. Простейшая версия алгоритма учитывает лишь малую часть этих корреляций. Для того чтобы учесть все корреляции, мы должны выполнить перебор всех путей от узла питания к рассматриваемому транзистору. Заметим, что для этой цели SP-BDD-представление является очень удобным.

Вторая проблема следует из того факта, что МОП-транзистор является двунаправленным. В простейшей версии алгоритма потенциал питания приходит к транзистору x только через его верхний узел. Но в действительности потенциал питания, приходящий к x через его нижний узел, также должен быть учтен.

Для пояснения этой проблемы рассмотрим *p*-транзистор *f* в КМОП-вентиле, показанном на рис. 2. Предположим, что корреляции между входными сигналами вентиля отсутствуют. Простейшая версия алгоритма полагает, что если транзистор *e* не проводит, то *f* не находится в напряженном состоянии. Но в действительности напряженное состояние *f* может быть также следствием проводящего состояния либо транзистора *d*, либо транзисторов *b* и *c*.

Простейшая версия алгоритма дает

$$Pstress(f) = PLZ(e) \cdot PLZ(f). \qquad (14)$$

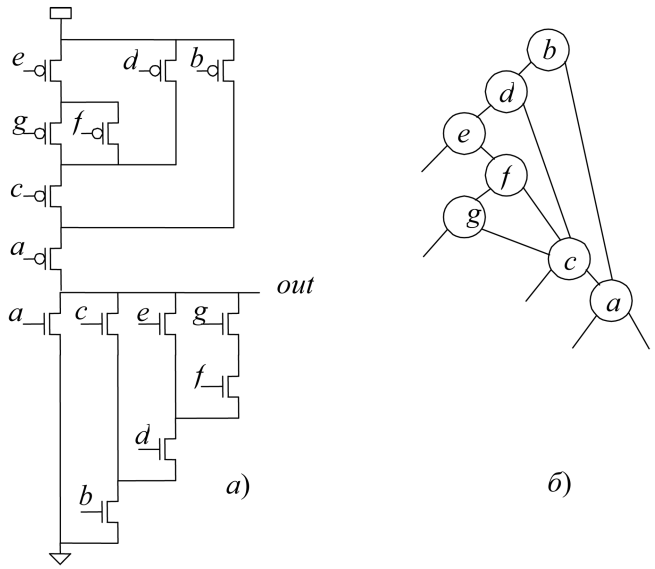


Рис. 2. КМОП-вентиль (*а*) и его представление в виде SP-BDD (*б*)

Однако точная формула выглядит так:

$$Pstress(f) = (1 - (1 - PLZ(e)) \cdot (1 - PLZ(d)) \times \times (1 - PLZ(b) \cdot PLZ(c))) \cdot PLZ(f). \quad (15)$$

Решение этих двух проблем делает алгоритм вычисления *Pstress* значительно более точным.

Алгоритм состоит из следующих этапов.

1. Выполнить полный перебор путей в верхней цепи КМОП-вентиля.

2. Для каждого транзистора верхней цепи повторить пп. 2.1, 2.2.

2.1. Сформировать полный список путей (напрягающих путей) от узла питания до данного транзистора.

2.2. Вычислить вероятность напряженного состояния для данного транзистора.

Ниже приведено более детальное описание этапов 1, 2.1, 2.2.

Полный перебор путей в верхней цепи. Для полного перебора путей используется SP-BDD-представление КМОП-вентиля. Следующая структура данных используется для вершины SP-BDD:

```
struct vertex {
    struct vertex*low,              /* 0-потомок                    */
                 *high,             /* 1-потомок                    */
                 *next;             /* вершина со след.
                                       значением индекса            */
    int          index;
    double       PLZ;               /* вероятность ло-
                                       гического 0 на за-
                                       творе транзистора            */
    double       SC[ ];             /* коэффициенты
                                       корреляции                   */
    path_list    list_of_paths;     /* список путей от
                                       узла питания к
                                       верхнему узлу тр-ра          */
    path_list    list_of_stressing_paths; /* см. ниже               */
    ...          /* другие поля     */                              */
};
```

Кроме этого, следующие данные относятся к КМОП-вентилю в целом:

```
struct vertex *root;            /* корневая вершина SP-BDD        */
path_list     list_of_full_paths; /* пути от узла питания до
                                     выхода вентиля                */
```

Алгоритм полного перебора путей может быть описан с помощью следующего псевдокода, в котором путь — это последовательность вершин SP-BDD (т. е. последовательность транзисторов):

```
for (vert = каждая вершина SP-BDD)
    vert- > list_of_paths пустой;
list_of_full_paths пустой;
vert = root;
добавить пустой путь к vert- > list_of_paths;
while (vert ! = NULL) {
    if (vert- > low- > list_of_paths пуст)
        vert- > low- > list_of_paths = vert- > list_of_paths;
    for (каждого пути из vert- > list_of_paths {
        current_path = path;
        добавить vert в current_path;
        if(vert- > high! = NULL)
            добавить current_path в vert- > high- > list_of_paths;
        else
            добавить current_path в list_of_full_paths;
    }
    vert = vert- > next;
}
```

Формирование полного списка напрягающих путей. Напрягающий путь транзистора x — это путь от узла питания к x (к его либо верхнему, либо нижнему узлу). Транзистор x также включается в каждый из его напрягающих путей. Если напрягающий путь транзистора находится в проводящем состоянии, то транзистор находится в напряженном состоянии. Если все напрягающие пути транзистора находятся в непроводящем состоянии, то транзистор не находится в напряженном состоянии. Формирование списков напрягающих путей можно описать следующим псевдокодом;

```
for (vert = каждая вершина SP-BDD)
  vert- > list_of_stressing_paths пустой;
vert = root;
while (vert ! = NULL) {
    if (vert- > list_of_paths содержит единственный путь,
    и этот путь пустой) {
        current_path = path;
        добавить vert в current_path;
        добавить current_path в vert- > list_of_stressing_paths;
    }
    else {
        for (каждого пути из vert- > list_of_paths) {
            current_path = path;
            добавить vert в current_path;
            добавить current_path в vert- > list_of_stressing_ paths;
        }
        vert1 = vert;
        while (vert1 ! = NULL) {
            vert1 = vert1- > next;
            for (каждого пути из vert1- > list_of_paths, со-
            держащего vert) {
                path1 = path;
            for (каждого пути из vert1- > list_of_paths, не со-
            держащего vert) {
                path2 = path;
                current_path = часть пути pathl,
                начинающаяся с vert;
                добавить path2 в current_path;
                добавить current_path в vert- > list_of_stressing_
                paths;
            }
        }
```

```
    }
    for (каждого пути из list_of_full_paths, содержащего
    vert) {
      path1 = path;
      for (каждого пути из list_of_full_paths, не содержа-
      щего vert) {
        path2 = path;
        current_path = часть пути path1, начинающаяся
        с vert;
        добавить path2 в current_path;
        добавить current_path в vert- > list_of_stressing_paths;
      }
    }
  }
почистить vert->list_of_stressing_paths; /* удалить повто-
ряющиеся копии /*
  vert = vert- > next;
}
```

Вычисление вероятности напряженного состояния. Вероятность напряженного состояния вычисляется для каждого транзистора верхней цепи вентиля на основе его списка напрягающих путей. Пусть символы типа a, b, c, ... используются для обозначения как транзисторов, так и событий нахождения этих транзисторов в проводящем состоянии. Рассмотрим следующее сложное событие: a & b & c. Если мы учитываем только парные корреляции, то мы можем использовать следующую аппроксимацию для сложного события:

$$p(a\&b\&c) = (p(a/b)p(b/c)p(c/a)) = \\ = \frac{p(a\&b)p(b\&c)p(c\&a)}{p(a)p(b)p(c)} = \\ = p(a)p(b)p(c)SC^{ab}SC^{bc}SC^{ac}, \quad (16)$$

где $p(a/b)$ — вероятность a при условии b; $SC^{ab} = SC_{00}^{ab}$ — коэффициент корреляции.

Рассмотрим более общую ситуацию: t — путь в верхней цепи КМОП-вентиля. Вероятность проводящего состояния t:

$$p(t) = p\left(\prod_{k=1}^{n} a_k\right) = \prod_{k=1}^{n} p(a_k) \prod_{1 \le k \le l \le n} SC^{a_k a_l}. \quad (17)$$

Пусть теперь $T = \{t_i, i = 1, ..., m\}$ — список напрягающих путей для транзистора x, $t_i = (a_{i1}, ..., a_{in_i})$ — путь, $M = \{1, ..., m\}$ — множество индексов. Вероятность напряженного состояния для транзистора x:

$$Pstress(x) = p(T) = 1 - p\left(\prod_{i=1}^{m} \bar{t}_i\right) \approx \\ \approx \sum_{i \in M} p(t_i) - \sum_{\substack{i,j \in M \\ i<j}} p(t_i)p(t_j)SC^{t_i t_j}, \quad (18)$$

где

$$p(t_i) = \prod_{k=1}^{n_i} p(a_{ik}) \prod_{1 \le k \le l \le n_i} SC^{a_{ik}a_{il}}, \quad (19)$$

$$SC^{t_i t_j} = \prod_{\substack{1 \le k \le n_i \\ 1 \le l \le n_j}} SC^{a_{ik}a_{jl}}. \quad (20)$$

Отметим, что если некоторый транзистор a содержится как в пути t_i, так и в пути t_j (см. формулу (20)), то нужно использовать следующее соотношение:

$$SC^{aa} = \frac{p(a\&a)}{p(a)p(a)} = \frac{1}{p(a)}. \quad (21)$$

3. Результаты численных экспериментов

Для проверки важности учета корреляций между сигналами и потенциала питания, приходящего через нижний узел транзистора, были выполнены следующие численные эксперименты.

Были сгенерированы *PLZ* и парные корреляции между сигналами для большой комбинационной схемы, взятой из стандартного набора ISCAS-85. Для всех первичных входов *PLZ* инициализировалась значением 0,5, а SC_{00} для каждой пары первичных входов инициализировался значением 1 (что означает отсутствие корреляции). Затем *PLZ* и SC_{00} были распространены по схеме. *PLZ* была вычислена для выхода каждого вентиля, а SC_{00} были вычислены для набора входов каждого вентиля.

После этого в схеме были выбраны некоторые вентили, и для каждого p-транзистора этих вентилей была вычислена вероятность напряженного состояния. Первый пример выбранного вентиля — OAI21, он показан на рис. 3.

Табл. 1 и 2 показывают результаты вычисления *Pstress* для двух вариантов вентиля OAI21. В этих таблицах "корр" означает "корреляции", "нижн" — "нижний узел транзистора", "+" — "с учетом" и "—" — "без учета".

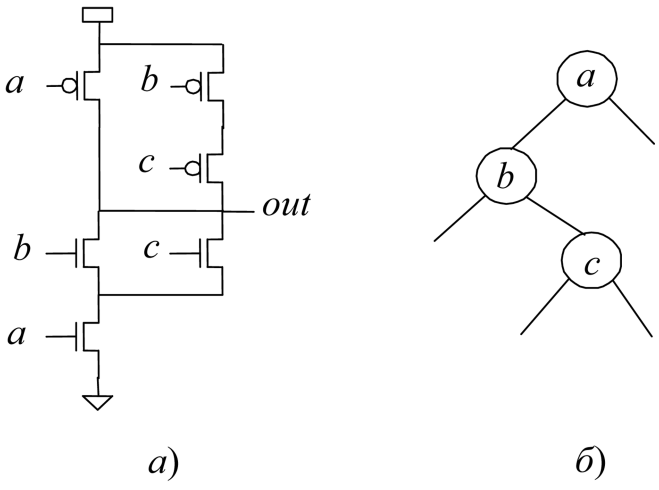


Рис. 3. Вентиль OAI21 (*а*) и его представление в виде SP-BDD (*б*)

Таблица 1

Результаты вычисления *Pstress* для AOI21-a

Транзистор	*PLZ*	*Pstress*			
		корр− нижн−	корр− нижн+	корр+ нижн−	корр+ нижн+
a	0,6	0,6	0,6	0,6	0,6
b	0,25	0,25	0,25	0,25	0,25
c	0,5	0,125	0,35	0,029	0,407

Таблица 2

Результаты вычисления *Pstress* для AOI21-b

Транзистор	*PLZ*	*Pstress*			
		корр− нижн−	корр− нижн+	корр+ нижн−	корр+ нижн+
a	0,94	0,94	0,94	0,94	0,94
b	0,28	0,28	0,28	0,28	0,28
c	0,5	0,14	0,48	0,0	0,44

Таблица 3

Результаты вычисления *Pstress* для сложного вентиля (рис. 2)

Транзистор	*PLZ*	*Pstress*			
		корр− нижн−	корр− нижн+	корр+ нижн−	корр+ нижн+
a	0,53	0,30	0,30	0,37	0,37
b	0,55	0,55	0,55	0,55	0,55
c	0,27	0,17	0,19	0,14	0,14
d	0,75	0,75	0,75	0,75	0,75
e	0,26	0,26	0,26	0,26	0,26
f	0,87	0,23	0,57	0,18	0,51
g	0,67	0,17	0,41	0,24	0,26

Второй пример выбранного вентиля — это вентиль, показанный на рис. 2. Результаты для этого вентиля даны в табл. 3.

Из приведенных таблиц видно, что для некоторых транзисторов *Pstress* в сильной степени зависит от учета корреляций и сигнала через нижний узел (транзистор *c* в OAI21, транзисторы *a*, *f*, *g* в сложном вентиле). Вместе с тем, *Pstress* для транзисторов, непосредственно присоединенных к узлу питания, всегда равен их *PLZ* (транзисторы *a*, *b* в OAI21, транзисторы *b*, *d*, *e* в сложном вентиле). Поэтому можно сделать вывод, что учет корреляций между сигналами и потенциала питания, подаваемого на нижний узел транзистора, необходим для корректного вычисления вероятности напряженного состояния (и, следовательно, для корректного вычисления НОСТ-деградации быстродействия).

Заключение

В настоящей работе корректное вычисление вероятности напряженного состояния для *p*-канальных МОП-транзисторов в цифровых КМОП-схемах рассмотрено в качестве независимой (самостоятельной) проблемы. Решение этой проблемы необходимо для проведения точного анализа НОСТ-деградации.

Во-первых, разработан метод расчета вероятностей значения сигнала и парных коэффициентов корреляции между сигналами. Этот метод в его модификации, предназначенной для анализа НОСТ-деградации, не требует больших вычислительных затрат и имеет сложность, близкую к линейной (по отношению к размеру схемы).

Во-вторых, предложен оригинальный корректный алгоритм вычисления вероятности напряженного состояния. В противоположность "наивному" подходу (учитывающему только пути от узла питания до верхнего узла *p*-транзистора) предлагаемый алгоритм учитывает как все парные корреляции между сигналами, так и появление потенциала питания через нижний узел транзистора. Алгоритм использует BDD-представление КМОП-вентиля. Результаты численных экспериментов подтверждают важность учета обоих этих факторов.

Дальнейшие исследования в этой области могут включать в себя расширение алгоритма вычисления вероятности напряженного состояния на случай не-ПП (ПП — последовательно-параллельных) верхних цепей КМОП-вентиля, а также на случай схем с элементами памяти (последовательностных схем, содержащих триггеры и защелки).

Список литературы

1. **Kumar Sanjay V. et al.** An Analytical Model for Negative Bias Temperature Instability // Proc. International Conference on Computer-Aided Design. 2006. P. 493—497.
2. **Reddy Vijay et al.** Impact of Negative Bias Temperature Instability on Product Parametric Drift // Proc. ITC—2004. P. 148—155.
3. **Kumar Sanjay V. et al.** NBTI-Aware Synthesis of Digital Circuits // Proc. Design Automation Conference. 2007. P. 370—375.
4. **Paul Bipul C. et al.** Impact of NBTI on the Temporal Performance Degradation of Digital Circuits // IEEE Electron Device Letters. 2003. P. 560—562.
5. **Chakravarthi S.** A Comprehensive Framework For Predictive Modeling of Negative Bias Temperature Instability // Proc. IRPS 2004. P. 273—282.
6. **Wang Wenping et al.** The Impact of NBTI on the Performance of Combinational and Sequential Circuits // Proc. Design Automation Conference. 2007. P. 364—369.
7. **Wang Wenping et al.** An Efficient Method to Identify Critical Gates under Circuit Aging // Proc. ICCAD—2007. P. 735—740.
8. **Marculescu R., Marculescu D., Pedram M.** Switching Activity Analysis Considering Spatiotemporal Correlations // Proc. ICCAD—1994. P. 294—299.
9. **Ercolany S., Favalli M., Damiani M. et al.** Testability Measures in Pseudorandom Testing // IEEE Trans. on CAD. 1992, V. 11. P. 794—800.
10. **Glebov A., Blaauw D., Jones L.** Transistor Reordering for Low Power CMOS Gates Using SP—BDD Representation // Intern. Symp. on Low. 1995. P. 161—166.
11. **Gavrilov S., Glebov A., Rusakov S. et al.** Fast Power Loss Calculation for Digital Static CMOS Circuits // Proc. European Design & Test Conf. 1997. P. 411—415.
12. **Glebov A., Gavrilov S., Blaauw D. et al.** Library-Less Synthesis for Digital Static CMOS Circuits // Proc. ICCAD—1997. P. 658—663.

УДК 004.414.2

В. Н. Ильин, д-р техн. наук,
Р. А. Гришин, аспирант,
Московский авиационный институт (ТУ),
e-mail: GrishinRA@gmail.com

Методика оценки основных параметров цифровых устройств на ПЛИС на ранних этапах проектирования с использованием двухуровневого макромоделирования

Изложена новая методика оценки основных параметров цифровых устройств (ЦУ) на программируемых логических интегральных схемах (ПЛИС) в ходе ранних этапов проектирования — этапов разработки технического предложения (ТП) и эскизного проекта (ЭП). Сформулированы требования, которым должна удовлетворять такая методика. Показано, что предлагаемая методика оценки с использованием двух уровней макромоделей соответствует предъявленным требованиям и позволяет рассмотреть большее число вариантов ЦУ, что сокращает сроки разработки ТП и ЭП и повышает качество проекта.

Ключевые слова: *макромодель, системное проектирование, ПЛИС.*

Развитие информационной техники в ходе двух последних десятилетий связано с широким использованием цифровых устройств (ЦУ), позволяющих одновременно достичь высоких технических характеристик и гибкости алгоритмов обработки информации [1]. В то же время условия рыночной экономики требуют ускоренной разработки ЦУ. Выполнение этих требований возможно на пути использования современной элементной базы в совокупности с новыми методами проектирования и новыми типами САПР. Ниже рассмотрены их особенности применительно к проектированию ЦУ.

Особенности современной элементной базы. Эпоха массового использования в ЦУ традиционной цифровой элементной базы (счетчиков, регистров, логических вентилей и т. п.) закончилась вследствие существенных недостатков этих элементов (малой степени интеграции, низкого быстродействия, больших габаритных размеров и потребляемой мощности) по сравнению с современными цифровыми программируемыми микросхемами (ЦПМС), к которым относятся микропроцессорная техника (микроконтроллеры, микропроцессоры, спецпроцессоры цифровой обработки сигналов) и программируемые логические интегральные схемы (ПЛИС) [2]. Применение ЦПМС позволяет не только устранить отмеченные недостатки, но и значительно сократить сроки разработки, так как с помощью ЦПМС можно изменять алгоритм работы устройства даже когда схема уже смонтирована на печатной плате. Благодаря этому стало возможным параллельно реализовывать устройство физически и разрабатывать алгоритм его работы.

Влияние современной элементной базы на маршрут проектирования ЦУ. Новая элементная база требует и новых подходов к проектированию. Традиционный маршрут проектирования [3] на дискретных элементах укрупненно состоит из этапов, указанных в табл. 1. Его особенностью является последовательная реализация этих этапов.

Для параллельной физической реализации устройства на современной элементной базе и разработки алгоритма его работы используется новый, более адекватный этой элементной базе маршрут проектирования (табл. 2). Важным отличием этого нового маршрута от общепринятого для дискретных элементов является то, что требования к аппаратной части ЦУ должны быть определены не после, а до разработки алгоритмов функционирования устройства, т. е. значительно раньше.

Современные ЦУ могут иметь в своем составе традиционные дискретные элементы, программируемую логику (ПЛИС), а также процессорное ядро. Требования к аппаратной части ЦУ основываются на выходных параметрах указанных трех составных частей ЦУ, определенных с учетом способа реализации ЦУ. Поэтому в обсуждаемом маршруте проектирования необходимо определить эти параметры уже на этапе подготовки технического проекта.

Таблица 1

Традиционный маршрут проектирования

Этап	Состав этапа
Подготовка технического предложения	Определение предложений по архитектуре Рассмотрение нескольких вариантов реализации функций системы
Эскизное проектирование	Разработка алгоритмов функционирования устройства. Определение основных требований к аппаратной части (быстродействие, объем памяти, внешние интерфейсы и т. д.). Конструкторско-технологическая проработка компоновки и трассировки схемы
Технический проект	Окончательная корректировка алгоритма. Уточнение требований к аппаратной части. Детальная схемотехническая и конструкторская проработка системы
Реализация и испытания	Физическая реализация и испытания

Таблица 2

Маршрут проектирования ЦУ с применением ЦПМС

Этап	Состав этапа	
Подготовка технического предложения	Определение предложений по архитектуре ЦУ	
	Рассмотрение нескольких вариантов реализации функций устройства. Разделение функций устройства между дискретными элементами, ПЛИС, а также процессорным ядром	
Эскизное проектирование	Определение основных требований к аппаратной части	
	Разработка алгоритмов функционирования для устройства в целом и всех его частей	Конструкторско-технологическая проработка компоновки и трассировки схемы
Технический проект	Уточнение требований к аппаратной части	
	Детальная проработка и реализация алгоритма	Детальная схемотехническая и конструкторская проработка устройства
Реализация и испытания	Физическая реализация. Испытания и окончательная корректировка алгоритма в случае необходимости	

Рассмотрим задачу определения внутренних и выходных параметров части устройства, реализуемой на ПЛИС. Важнейшими из них являются:

- количество логических ресурсов ПЛИС, необходимое для реализации алгоритма;
- необходимое быстродействие ПЛИС;
- необходимый объем памяти ПЛИС;
- число каналов связи ПЛИС с другими элементами схемы и внешней средой и их пропускная способность.

На сегодняшний день предполагаемые значения указанных параметров ЦУ определяются одним из трех способов:

- с использованием САПР аппаратно-программного (А/П) разделения;
- с использованием САПР, решающих задачи синтеза;
- эмпирическим.

Можно показать, что каждый из этих способов применительно к ранним этапам проектирования ЦУ имеет те или иные существенные недостатки: необходимость в отсутствующей исходной информации либо требование избыточной информации, ограниченность области применения, высокая трудоемкость, наличие значительного опыта проектирования и др. В связи с этим возникает необходимость в разработке новой методики оценки выходных параметров части ЦУ, реализуемой на ПЛИС (ПЛИС-части).

Новая методика оценки параметров

Новая методика должна удовлетворять следующим требованиям:

- возможность оценки выходных параметров ПЛИС-части ЦУ по минимальному набору исходных данных, которые должны быть известны уже на ранних этапах проектирования. Это позволит сформулировать требования к аппаратной части уже в начале работы над эскизным проектом и реализовать маршрут проектирования ЦУ с применением ЦПМС;
- упрощение структуры и сокращение числа исходных данных по сравнению с существующими методами;
- учет в процессе оценки архитектурных особенностей ПЛИС различных серий в целях получения достоверных результатов;
- меньшая трудоемкость получения оценки по сравнению с существующими методами.

В ходе развития методов проектирования ЦУ на СБИС задача упрощения структуры и сокращения числа исходных данных возникала неоднократно [4, 5]. Чаще всего ее решали путем перехода к более высокому уровню абстракции при описании устройства. Этим в основном был обусловлен переход от описания на логическом уровне (логические вентили, триггеры) к языкам описания аппаратуры на уровне RTL (регистры, счетчики, сумматоры, дешифраторы) [4] и позже к описанию с помощью сложнофункциональных блоков (СФ-блоков) типа цифровых фильтров, блоков деформации изображения, кодирования/декодирования и т. д.) [5, 6].

Используемый на сегодняшний день способ описания аппаратуры с помощью СФ-блоков также не лишен недостатков. Современные СФ-блоки достаточно сложны, имеют большое число входов/выходов, характеризуются множеством параметров (несколько десятков), большим разнообразием блоков, а стандарты в данной области развиты недостаточно [6]. Поэтому использование таких блоков для проектирования требует значительного опыта и времени. Кроме того, применение нового, более высокого уровня описания — уровня СФ-блоков — требует разработки методов его автоматизированного преобразования в описание более низкого уровня, используемого в имеющихся САПР. Тем не менее, несмотря на отмеченные недостатки, в новой методике предлагается все же использовать уровень СФ-блоков для описания проектируемого устройства, но при этом следует:

- значительно сократить число параметров блоков, ограничившись 2—4 важнейшими;
- полное описание структуры ЦУ заменить перечнем СФ-блоков с указанием основного пути прохождения (обработки) информации. В процессе проектирования такой перечень можно получить значительно раньше, чем полное структурное описание устройства.

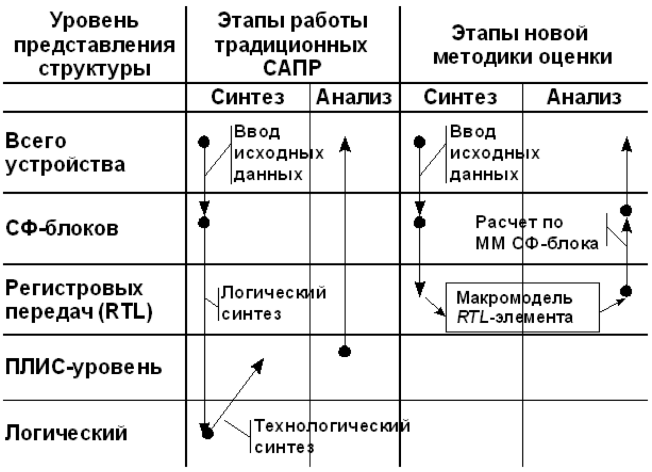


Рис. 1. Этапы традиционной и новой методик оценки параметров

В соответствии с требованиями к новой методике необходимо снизить трудоемкость оценки выходных характеристик ЦУ и обеспечить возможность работы с описанием в виде перечня СФ-блоков. Одним из методов сокращения вычислительных затрат при моделировании радиоэлектронных устройств является применение макромоделей [7, 8]. На рис. 1 показаны этапы процедур получения оценки выходных характеристик ЦУ традиционным методом — с помощью синтезирующих САПР и предлагаемым методом — посредством двух уровней макромоделей (ММ). Как видим, второй способ по числу этапов проектирования и соответственно затратам времени значительно экономичнее первого.

Макромодели можно классифицировать по степени сложности моделируемой подсхемы (модель СФ-блока, модель элемента уровня RTL и т. д.) и по степени детализации описания объекта моделирования (модели типа "черный", "серый", "белый" ящик). Предлагается разработать макромодели, позволяющие оценить предполагаемые выходные параметры моделируемой подсхемы с учетом ее реализации на ПЛИС. Для построения макромодели необходимо определить, во-первых, оптимальный уровень сложности моделируемых подсхем, для которого целесообразно макромоделирование (СФ-блоки, компоненты ПЛИС, элементы логического уровня или уровня RTL), и, во-вторых, степень детализации описания объекта моделирования, для которого строится макромодель.

С точки зрения выбора оптимального уровня сложности макромоделируемых подсхем можно отметить следующее. Создание макромоделей СФ-блоков, хотя и затруднено достаточно высокой сложностью и большим разнообразием элементов этого уровня, но оправдано их эффективностью, если удается сократить число описывающих их параметров. Хотя СФ-блоки и многообразны, но для отдельной прикладной области построение достаточного множества макромоделей возможно. В настоящее время проектирование ЦУ на ПЛИС и СБИС с использованием СФ-блоков получает все большее распространение [5, 6]. Помимо СФ-блоков в качестве объектов макромоделирования более низкого уровня целесообразно выбрать элементы уровня RTL (RTL-элементы), поскольку они:

- практически инвариантны к области применения устройства;
- менее чувствительны к способу соединения, чем элементы логического уровня, так как являются законченными функциональными единицами;
- их число невелико, что позволяет создать библиотеку макромоделей для всех элементов данного уровня.

Таким образом, целесообразно создать библиотеки макромоделей нижнего уровня для RTL-элементов и верхнего уровня для СФ-блоков. Базисом для построения структуры макромоделей верхнего уровня будут макромодели нижнего уровня, что упростит задачу макромоделирования СФ-блоков. Ниже описаны процедуры создания макромоделей двух указанных уровней.

А. Получение макромоделей нижнего уровня $Y1 = \Omega(X1)$.

1. Определить множество RTL-элементов, для которых необходимо построить макромодель.

2. Для каждого такого RTL-элемента построить ряд макромоделей разной структуры типа "черный ящик", различающихся требуемыми входными и выходными данными и структурой.

2.1. Задать структуру макромодели.

2.2. Задать множество входных $X1$ и выходных $Y1$ параметров макромодели.

2.3. Используя специализированную САПР, методом многовариантного анализа получить множество вариантов значений выходных параметров.

2.4. Обработать полученные в п. 2.3 данные, с помощью специальных САПР (MATLAB, MATHCAD и т. п.) и получить функциональную зависимость $Y1 = \Omega(X1)$ либо таблицу значений, отражающую эту зависимость, которые и составят макромодель элемента RTL-уровня.

3. Объединить полученные макромодели нижнего уровня в библиотеку.

Б. Получение макромоделей верхнего уровня $Y2 = \Phi(Y1, X2)$ для СФ-блоков.

1. Определить множество СФ-блоков, для которых необходимо построить макромодели.

2. Для каждого СФ-блока составить структуру его макромодели, элементами которой должны быть RTL-элементы. Эта структура должна быть

проста настолько, чтобы ее можно было рассматривать как функцию $Y2 = \Phi(Y1, X2)$.

2.1. Задать структуру макромодели СФ-блока в базисе RTL-элементов.

2.2. Задать множество входных $X2$ и выходных $Y2$ параметров СФ-блока. В множество $X2$ входят как независимые параметры $X2_{НЗ}$, так и зависимые (структурные) $X2_З$, для вычисления которых необходимо определить функцию $X2_З = \Phi_{ВХ}(X2_{НЗ})$.

2.3. Определить функцию $X1 = \Phi_{RTL}(X2)$ расчета входных параметров $X1$ всех RTL-элементов, вошедших в структуру данного СФ-блока.

2.4. Составить для определенной в п.2.1 структуры макромодели конкретную функцию $Y2 = \Phi(Y1, X2)$. Функция $\Phi(Y1, X2)$ в совокупности с функциями $\Phi_{ВХ}(X2_{НЗ})$ и $\Phi_{RTL}(X2)$ составят макромодель СФ-блока. Для каждого СФ-блока функции Φ, $\Phi_{ВХ}$, Φ_{RTL} уникальны и формализовать их построение практически невозможно, так как они зависят от конкретной структуры макромоделируемого СФ-блока и для их построения необходимо анализировать эту структуру. Значительные усилия, необходимые для создания макромодели СФ-блока, оправдываются только при условии многократного использования этих макромоделей в дальнейшем, поэтому необходимо уделять достаточное внимание вопросу выбора моделируемых СФ-блоков (п. 1).

2.5. Вернуться к п. 2.1 и задать новый вариант структуры СФ-блока с новым составом RTL-элементов.

3. Объединить полученные макромодели верхнего уровня в библиотеку СФ-блоков типа "серый ящик".

Аналогично, для расчета выходных параметров всего ЦУ необходимо для каждого выходного параметра $Y3$ этого ЦУ разработать формульную зависимость $Y3 = \Psi(Y2)$, позволяющую рассчитать выходные параметры ЦУ по известным значениям выходных параметров СФ-блоков. Предполагается, что эти зависимости предельно просты (сумма, минимум, максимум и т. п.). Описанная методика двухуровневого макромоделирования иллюстрируется рис. 2.

Создание макромоделей двух уровней выполняется разработчиком этой методики, а не ее пользователем. Пользователь методики (разработчик ЦУ), имея полученные описанным способом библиотеки макромоделей, сможет быстро оценить выходные параметры ЦУ, рассчитать большое число вариантов ЦУ, различающихся набором составляющих их СФ-блоков, структурой СФ-блоков и значениями их входных параметров. Для эффективного использования данной процедуры необходимо разработать САПР, реализующую ее. Такая САПР должна включать в себя подготовленные ранее библиотеки макромоделей СФ-блоков и RTL-элементов. САПР позволит разработчику ЦУ оценить выходные характеристики проектируемого устройства без использования трудоемких процедур логического и технологического синтеза, размещения и трассировки, которые на сегодняшний день являются практически единственным способом получения более-менее достоверной оценки выходных характеристик достаточно сложного устройства.

Оценка выходных параметров ЦУ выполняется в соответствии с процедурой В, которую иллюстрирует рис. 2.

Процедура В. Оценка выходных параметров ЦУ.

1. Покрыть ЦУ набором СФ-блоков (верхняя часть рис. 2) и задать их независимые входные параметры ($X2_{НЗ}$).

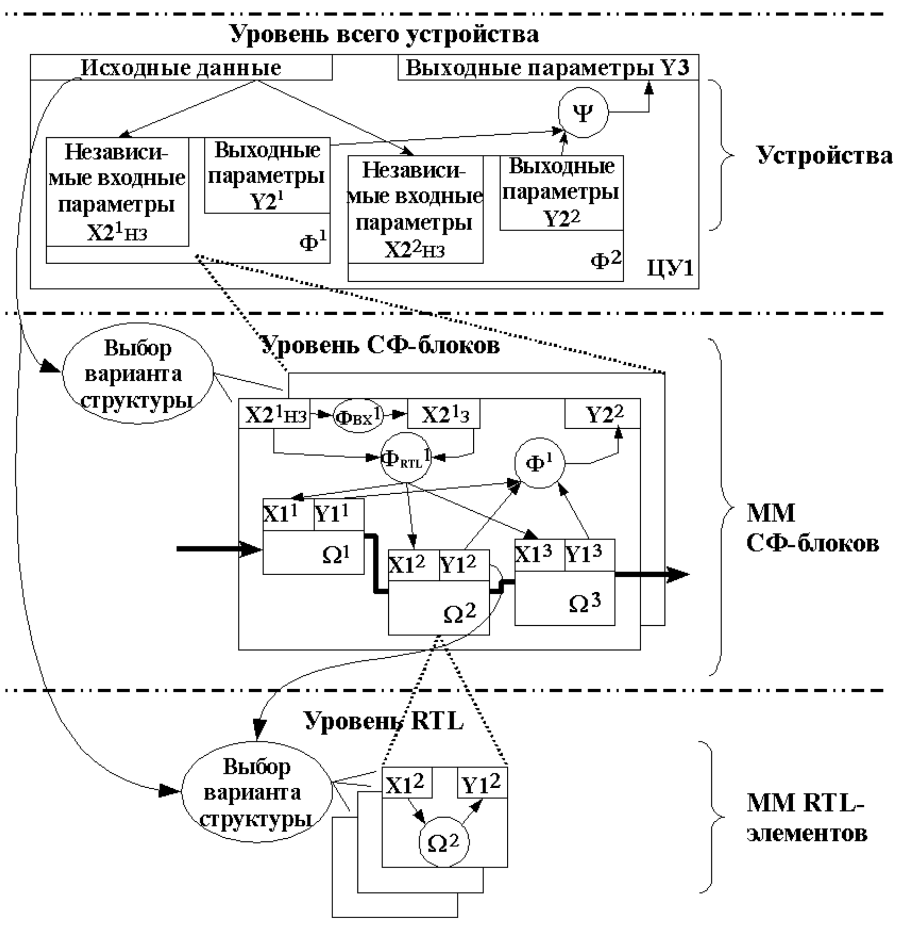


Рис. 2. Схема оценки выходных параметров ЦУ

2. Для каждого СФ-блока, вошедшего в ЦУ, выполнить следующее:

2.1. Выбрать один из вариантов структуры СФ-блока из библиотеки макромоделей. Наиболее подходящий вариант структуры СФ-блока определяется исходя из значений входных параметров $X2$ и заданных пользователем критериев (минимизация затрат логических элементов, памяти и т. д.).

2.2. Используя функцию $\Phi_{вх}(X2_{нз})$, определить значения структурных параметров $X2_з$ СФ-блока.

2.3. Определить значения входных параметров $X1$ RTL-элементов, вошедших в структуру данного СФ-блока, используя функцию $\Phi_{RTL}(X2)$.

3. Определить выходные параметры $Y1$ RTL-элементов, используя макромодель нижнего уровня $Y1 = \Omega(X1)$ и полученные значения входных параметров $X1$.

4. Рассчитать выходные параметры $Y2$ СФ-блоков, используя полученные на предыдущем шаге оценки выходных параметров RTL-элементов ($Y1$) и макромодель верхнего уровня $Y2 = \Phi(Y1, X2)$.

5. Используя функцию $\Psi(Y2)$, оценить выходные параметры ЦУ в целом ($Y3$).

Обозначения на рис. 2: окружность — та или иная функция, обозначение которой изображено внутри окружности (например, Ω^2 в окружности — функция $Y1 = \Omega(X1)$, составляющая макромодель RTL-элемента № 2); Φ^j вне окружности обозначает макромодель j-го СФ-блока, $X2^j_{нз}$ и $X2^j_з$ — это соответственно множества его независимых и зависимых параметров, а $Y2^j$ — множество его выходных параметров; Ω^j вне окружности — макромодель j-го RTL-элемента, $X1^j$ и $Y1^j$ — соответственно множества его входных и выходных параметров. Тонкими стрелками обозначена передача параметров, толстыми стрелками — передача данных.

Пример применения новой методики

Создание макромодели RTL-элементов. Задачу построения макромодели нижнего уровня рассмотрим на примере одного RTL-элемента — умножителя. Будем считать, что существует лишь один вариант структуры умножителя. В этом случае в множество его входных параметров $X1$ войдут: разрядность множимого (m) и множителя (n), способ представления операндов (со знаком или без). Умножитель характеризуется двумя выходными параметрами $Y1$: необходимое число логических элементов (ЛЭ) L_{mult}; время срабатывания t_{mult}.

Получим зависимость времени срабатывания умножителя t_{mult} от разрядности его операндов (m, n), считая, что $m = n = W$. Экспериментально, т. е. используя специальную САПР ПЛИС — Quartus II, получим значения времени срабатывания t_{mult}, соответствующие различной разрядности операндов W. На рис. 3 показаны экспериментальные точки и их линейная аппроксимация $t_{mult} = \Omega(W)$.

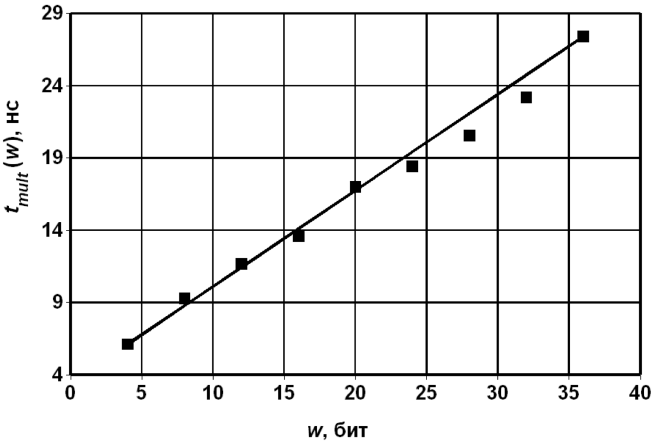


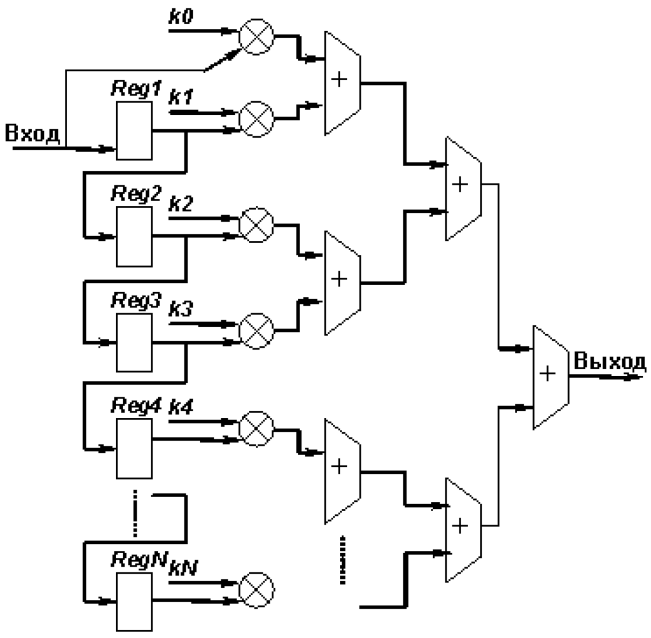
Рис. 3. Зависимость времени срабатывания t_{mult} умножителя от разрядности его операндов W (для ПЛИС фирмы Altera серии Cyclone-II)

Точность определения выходных параметров RTL-элемента зависит от числа экспериментальных точек и их распределения в пространстве параметров. В данном примере ошибка определения времени срабатывания t_{mult} не превысила 5,5 %.

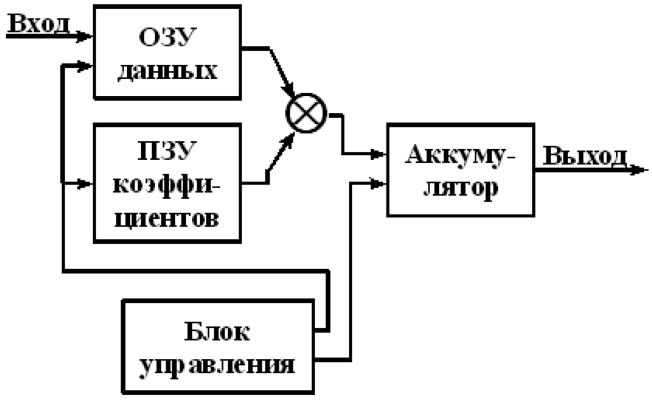
Полученная зависимость войдет в макромодель умножителя $Y1 = \Omega(X1)$. Аналогичным образом получают зависимости и для других выходных параметров RTL-элемента. Семейство таких зависимостей и составит макромодель умножителя. В этом случае макромодель не содержит информации о принципе работы или структуре RTL-элемента и поэтому является "черным ящиком".

Создание макромодели СФ-блоков. В качестве примера рассмотрим один СФ-блок — цифровой фильтр с конечно-импульсной характеристикой (КИХ-фильтр) Можно выделить два основных варианта структуры цифровой фильтрации: параллельный и конвейерный (рис. 4). Процесс цифровой фильтрации заключается в свертке задержанных отсчетов входного сигнала с массивом коэффициентов. В фильтре параллельной структуры умножение на все коэффициенты происходит одновременно, в конвейерном — последовательно. Параллельный вариант цифровой фильтрации требует больше аппаратуры, но позволяет обрабатывать данные с более высокой частотой.

Далее будем рассматривать цифровую фильтрацию с параллельной структурой. Как видно из рис. 4, *а*, в состав фильтра войдут следующие RTL-элементы: блоки памяти, умножители, сумматоры. Число RTL-элементов каждого типа для данного СФ-блока входит в множество зависи-



а)



б)

Рис. 4. Варианты структуры цифрового фильтра:
а —параллельного, *б* — конвейерного

мых параметров $X2_з$ и определяется значениями независимых входных параметров, например:

$$X2_з^1 = N + 1, \quad (1)$$

где $X2_з^1$ — первый зависимый параметр КИХ-фильтра — число умножителей в структуре; N — независимый параметр КИХ-фильтра — число элементов в линии задержки.

Таким образом, формула (1) войдет в множество $\Phi_{вх}$.

Для расчета значений входных параметров $X1$ рассматриваемых RTL-элементов необходимо представить эти параметры как функцию входных параметров СФ-блока: $X1 = \Phi_{RTL}(X2)$. Так, например, разрядность операндов при умножении определяется как

$$m^1 = W_{data}; \quad (2)$$

$$n^1 = W_{koef}, \quad (3)$$

где m^1 и n^1 — разрядность множимого и множителя для 1-го RTL-элемента (умножителя); W_{data}, W_{koef} — независимые входные параметры КИХ-фильтра — разрядность данных и коэффициентов соответственно. Формулы (2) и (3) войдут в множество $\Phi_{RTL}(X2)$.

Далее, составим выражения для расчета значений выходных параметров СФ-блока по известным значениям выходных параметров макромодели низкого уровня и входных параметров RTL-элементов: $Y2 = \Phi(Y1, X2)$. Для некоторых выходных параметров СФ-блоков можно указать общий метод их расчета, не зависящий от конкретного СФ-блока. Так, например, число логических элементов (ЛЭ), необходимое для реализации СФ-блока ($L_{СФ}$) равно сумме ЛЭ, необходимых для построения всех RTL-элементов, входящих в данный СФ-блок. То же справедливо и для памяти ($M_{СФ}$):

$$L_{СФ} = \Sigma L_i;$$

$$M_{СФ} = \Sigma M_i,$$

где L_i и M_i — соответственно число ЛЭ и объем памяти, необходимые для реализации i-го RTL-элемента

Конкретные формулы для расчета L и M определяются структурой СФ-блока, построенной в базисе RTL-элементов. Методики расчета таких выходных параметров, как число линии связи и их пропускная способность, разрабатываются для каждого СФ-блока индивидуально.

В качестве примера приведем формулу расчета $L_{СФ}$ для фильтра с параллельной структурой, обозначив $L_{СФ}$ как L_{FIR} ($L_{FIR} \in Y2$):

$$L_{FIR} = NL'_{reg}(W_{data}) + (N+1)L'_{reg}(W_{koef}) + (N+1)L'_{mult}(W_{data}, W_{koef}) + \sum_{k=1}^{K}(L'_{sum}) + L'_{reg}(W_{data} + W_{koef} + K),$$

где L'_{reg}, L'_{mult}, L'_{sum} — число ЛЭ, необходимое для реализации регистра, умножителя и сумматора соответственно при заданной разрядности данных и операндов. Здесь и далее штрихом обозначены функции, составляющие макромодель RTL-элемента, т. е. $A' \in \Omega$, где A — обозначение выходного параметра RTL-элемента.

Таким образом, нами подготовлены необходимые макромодели RTL-элементов (нижний уровень) и СФ-блоков (верхний уровень). На основе полученных макромоделей была выполнена оцен-

Таблица 3

Результаты эксперимента по применению предложенной методики для оценки выходных параметров КИХ-фильтра

Входные параметры			Выходные параметры							
			Фактические значения		Расчетные значения				Ошибка определения	
W_{data}	W_{koef}	N	L_{FIR}, ЛЭ	t_{FIR}, нс	L_{FIR}, ЛЭ	t_{FIR}, нс	C_{FIR}, шт.	T_{FIR}, Кбит/с	L_{FIR}, %	t_{FIR}, %
16	16	3	610	14,131	664	14,63	32	1408	−8,85	−3,54
16	16	7	1283	17,975	1368	17,83	32	1408	−6,63	0,83
16	16	9	1595	19,981	1730	17,83	32	1408	−8,46	10,79
26	26	3	1399	18,274	1388	19,19	52	2288	0,79	−5,03
26	26	7	2967	22,319	2836	22,39	52	2288	4,42	−0,31
26	26	9	3701	23,922	3570	22,39	52	2288	3,54	6,41
32	32	3	1932	19,713	1880	19,99	64	2816	2,69	−1,43
32	32	7	4092	23,535	3832	23,19	64	2816	6,35	1,47
32	32	9	5158	25,523	4818	23,19	64	2816	6,59	9,15

ка выходных параметров ЦУ в соответствии с процедурой В. В табл. 3 приведены фактические значения выходных параметров ЦУ, состоящего из одного СФ-блока — КИХ-фильтра, полученные путем синтеза цифровой фильтрации (ЦФ) в САПР Quartus II для ПЛИС серии Cyclone-II, и значения параметров, рассчитанные в соответствии с предложенной методикой для ряда наборов входных данных. Фактическое число линии связи и пропускная способность не указаны, так как их расчетные значения точно соответствуют фактическим. В таблице обозначены: $L_{FIR} = L_{СФ}$ для ЦФ; t_{FIR} — время срабатывания ЦФ; C_{FIR} — число линий связи; T_{FIR} — общая пропускная способность линий связи.

Как видно из таблицы, ошибка определения затрат ЛЭ достигает 8 %, а времени срабатывания — 10 %, что связано с упрощениями, принятыми при создании макромоделей верхнего уровня (как было отмечено ранее, точность определения параметров макромоделей RTL-элементов выше). Тем не менее, 10 %-ная точность, по мнению ряда специалистов, является удовлетворительной для ранних этапов проектирования (см., например, [9, 10]). Кроме того, по абсолютной величине расчетные и фактические значения различаются незначительно и остаются достаточно информативными с точки зрения оценки различных вариантов реализации на ранних этапах проектирования.

Выводы

Основные результаты работы сводятся к следующему.

- Сформулирован перечень требований к новой методике оценки выходных параметров ЦУ, включающий:
- минимальность набора исходных данных, которые должны быть известны уже на ранних этапах проектирования;
- существенное упрощение способа описания устройства по сравнению с существующими САПР;
- учет архитектурных особенностей ПЛИС различных серий в целях получения достоверных результатов.
- Предложена и сформулирована концепция новой методики, включающая:
- ограничение состава исходных данных перечнем СФ-блоков устройства с указанием минимального числа их параметров, что позволяет выполнить оценку по минимальному набору исходных данных;
- использование двух уровней макромоделей: нижнего уровня для элементов регистрового уровня и верхнего уровня для СФ-блоков. Макромодели нижнего уровня строятся методом "черного ящика" путем обработки экспериментальных данных, а макромодели верхнего уровня — методом "серого ящика" с учетом внутренней структуры каждого СФ-блока.
- В рамках предложенной концепции разработана последовательность расчета выходных параметров макромоделей обоих уровней, учитывающая их взаимозависимость.
- Предложен упрощенный способ представления структуры всего устройства совокупностью СФ-блоков, структуры которых определяются методом структур-прототипов, и определения выходных параметров устройства на основе выходных параметров макромоделей СФ-блоков. Это позволяет решить поставленную задачу оценки выходных параметров плис-части ЦУ с приемлемой точностью без привлечения сложных алгоритмов прямого синтеза.
- Приведен пример оценки характеристик СФ-блока (цифровой фильтр с конечной импульсной характеристикой), подтверждающий конструктивность основных принципов, заложенных в предлагаемую методику.

Применение данной методики будет способствовать сокращению сроков разработки устройств на ПЛИС и снижению стоимости проектируемого изделия.

Список литературы

1. **Столингс В.** Структурная организация и архитектура компьютерных систем. 5-е изд. / Пер. с англ. М.: Изд. дом "Вильямс", 2002. 896 с.
2. **Грушвицкий Р. И., Мусаев А. Х., Угрюмов Е. П.** Проектирование систем на микросхемах с программируемой структурой. 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2006.
3. **Пронин Е. Г., Могуева О. В.** Проектирование бортовых систем обмена информации. М.: Радио и связь, 1989.
4. **Donlin M.** ASIC complexity fuses drive to HDL design // Computer Design, May, 1991.
5. **Moretti G.** Design Complexity Requires System-Level Design // EDN-Europe, April. 2005.
6. **Немудров В., Мартин Г.** Системы-на-кристалле. Проектирование и развитие. М.: Техносфера, 2004.
7. **Корячко В. П., Курейчик В. М., Норенков И. П.** Теоретические основы САПР: Учебник для вузов. М.: Энергоатомиздат, 1987.
8. **Ильин В. Н., Фролкин В. Т., Бутко А. И. и др.** Автоматизация схемотехнического проектирования: Учеб. пос. для вузов / Под ред. В. Н. Ильина. М.: Радио и связь, 1987. 368 с.
9. **Гуткин Л. С**. Проектирование радиосистем и радиоустройств: Учеб. пос. для вузов. М.: Радио и связь, 1986.
10. **Подиновский В. В., Ногин В. Д.** Парето-оптимальные решения многокритериальных задач. М.: Наука, 1982.

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

УДК 004.4.414

А. Ш. Сулейманов, канд. техн. наук,
Азербайджанский технический университет,
г. Баку, e-mail: akif@inbox.ru

Метод определения контекстных слов при анализе текста

Рассматривается модель текста произвольного содержания, которая позволяет отобразить семантическую связность и последовательность текста в виде структуры с использованием логических шкал, а также наглядно определить ключевые слова с контекстом, организовывать анализируемые тексты в единую, целостную информационную структуру, обеспечивая последующий анализ совокупности документов и выделение из них общих тематических кластеров.

Ключевые слова: *словосочетание, анализ, содержание, поиск, контекст, частотная модель, ключевые слова, текст, классификация.*

Введение

Задача анализа и поиска документов по содержанию предполагает решение двух основных задач: тематической классификации полнотекстовой информации; вычисления степени тематической принадлежности текста к заданному классу. Эти задачи связаны, прежде всего, с анализом текста, а именно, с анализом смыслового содержания текста, его тематической направленности. Предлагаемый подход к тематической классификации текстовой информации основывается на гипотезе о том, что словарный запас и частоты использования слов зависят от темы текста. В настоящее время данная гипотеза активно и успешно используется в тематико-ориентированных методах поиска.

Тематическая классификация предполагает выделение множества ключевых слов, определяющих тематику текста. При этом каждому из них приписывается вес, определяющий значимость данного слова в тематике, т. е. какие-то ключевые слова играют бóльшую роль в определении тематики, какие-то меньшую, но именно такая совокупность слов, с такой значимостью каждого из них в тематике и определяет тематическую направленность [1]. Данный подход обеспечивает снижение размерности решаемой задачи за счет перехода от основного текста к его представлению в виде множества ключевых слов, приближенно описывающих его содержание. Это необходимо, прежде всего, для последующей тематической идентификации сравниваемых текстов. Задача классификации в данном случае сводится к задаче отнесения текста к некоторому тематическому классу, описываемому множеством ключевых слов.

Постановка задачи. Опираясь на имеющиеся данные, а также на результаты собственных исследований, предлагается модель текста, учитывающая его связность и дополнения ключевых слов контекстом с использованием логических шкал (на основе методов сжатия данных).

Методы решения. Суть этого метода заключается в том, что тексты представляются в виде упорядоченного по частоте встречаемости слов словаря (W), составляющего содержание этих текстов и логических шкал, кодирующих позиции каждого слова в тексте (один из методов сжатия данных). Для этого сначала проводится статистический анализ текстов и определяется частота встречаемости каждого **значимого** слова (w — ин-

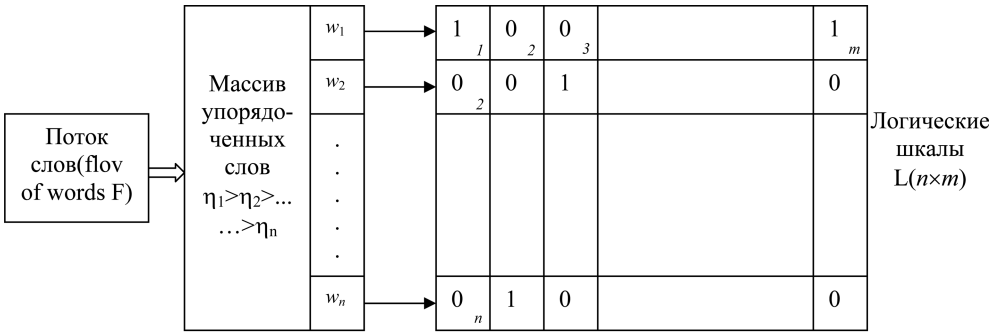


Рис. 1. Информационная структура текста

формационный элемент) без учета "стоп элементов" (предлоги, союзы, знаки препинания и т. д.), и каждый текст хранится в ассоциативной памяти для увеличения быстродействия во время поиска информации. Длина каждой логической шкалы равна числу всех слов текста в битах. При таком представлении структуры совокупность логических шкал образует матрицу **L** размером ($n \times m$), где n — число уникальных слов, m — число всех слов, составляющих текст (рис. 1). Позиция каждого слова отличается единицами в соответствующих логических шкалах. Число единиц в каждой шкале равно частоте встречаемости соответствующего слова в тексте. С помощью такой матрицы легко определить ключевые слова и их контекст. Контекст определяется с помощью соседних столбцов ключевых слов. Глубина контекста определяется числом столбцов, заданным экспертом. Так как уникальные слова в памяти размещаются в порядке убывания частоты встречаемости η, то легко определить пороговое значение количества ключевых слов.

Значит, суть данного подхода заключается в моделировании структуры текста информационным потоком и формировании этим потоком информационной структуры текста в виде матрицы размером $n \times m$.

Ключевые слова определяются по числу их вхождений в текст (числу единиц соответствующей **строки** логической шкалы), а именно, частота ключевых слов в тексте выше частоты других слов. В рамках рассматриваемой модели структурного представления текста это будет означать, что через логические шкалы данных слов чаще проходит информационный поток, и информационные элементы, соответствующие этим словам, имеют больше связей с другими информационными элементами.

Проблема заключается в определении порога, который отделяет ключевые слова от всех остальных. Очевидно, выбор порогового значения должен зависеть от конкретного текста, от таких характеристик модели как $d(D(W, C))_{\max}$, $d(D(W, C))_{\min}$ и $n(W)$ — числа уникальных слов в тексте.

Информационная структура $D(W, C)$ является совокупностью W — множества информационных элементов (множества строк логической шкалы) и C — набора связей между этими элементами ($D(W, C) \equiv F$);

C — набор связей между парами информационных элементов, который может содержать повторяющиеся связи в случае многократного прохождения информационного потока F через одни и те же пары элементов; $d(w) = n(C(w))$ — число пар связей в наборе $C(w)$; характеризует число связей данного информационного элемента с другими информационными элементами в структуре $D(W, C)$; $n(C(w))$ — равно числу повторений слова в тексте; $d(D(W, C))_{\max}$ — максимальное число пар связей информационного элемента (значимость данного слова в тематике текста) в информационной структуре $D(W, C)$; $d(D(W, C))_{\min}$ — минимальное число пар связей информационного элемента (w) в информационной структуре $D(W, C)$.

Однако этого недостаточно, чтобы корректно выделять тематику текста. Адекватность тематики (машинного представления темы текста в виде множества ключевых слов) по отношению к теме, которую для себя определяет человек после чтения текста, — вопрос открытый.

Выдвигается гипотеза о том, что корректное и адекватное машинное представление тематики текста должно включать в себя не только ключевые слова, но и контекст этих слов (словосочетаний), так как смысл любого слова определяется исключительно в контексте тех слов, которые употреблялись вместе с ним, близко, рядом по тексту. Сами по себе ключевые слова в отрыве от их контекста не отражают в полной мере тематическую направленность текста. Существующие исследования в психолингвистике подтверждают данный тезис [2].

Необходимость дополнения ключевых слов контекстом определяется также соображениями практического характера. Суть этих соображений заключается в следующем.

Особенности частотного распределения слов в тексте могут значительно затруднить выбор порогового значения и снизить качество последующего анализа документов на тематическую близость. Например, ситуация частотного выброса одного из слов. Непонятно при этом, какой необходимо устанавливать порог отсева, если частота повторений одного слова значительно превосходит все остальные, а все остальные при этом имеют оди-

наковую частоту. Значит надо либо устанавливать порог для выделения одного ключевого слова, или опускать порог и брать все слова текста в качестве тематики. Оба варианта неприемлемы — в одном случае тематика текста будет представлена в виде одного слова, в другом случае тематикой будут все слова. Организация последующего поиска тематически близких документов (текстов) на основе множества ключевых слов, выступающих в качестве поискового запроса, представляется в этом случае весьма проблематичной. Если поисковый запрос представлен одним словом — результат поиска может дать незначительное число тематически близких документов, если поисковый запрос представлен всеми словами документа, то результат поиска может дать слишком много "тематически далеких" документов (рассматривается вариант поиска — ''искать документы, содержащие хотя бы одно из слов запроса").

Дополнение ключевых слов контекстом в этом случае является вполне разумным и приемлемым вариантом решения данной проблемы. Общая последовательность метода будет выглядеть следующим образом.

1. Выделение множества всех информационных элементов, ранжированных по их степени $d(w)$ (числу повторений в тексте). Элемент с $d(D(W, C))_{\max}$ будет первым, и далее по убыванию.

2. Моделирование текста и формирование информационной структуры $D(W, C)$.

3. Выделение множества ключевых элементов.

Из множества всех информационных элементов (полученных на предыдущем этапе) берем n первых (n определяется на основе порогового значения), которые будут первичным множеством ключевых слов $M_p = \{k_1w_1, k_2w_2, ..., k_nw_n\}$. Коэффициенты $k_1, k_2, ..., k_n$ соответствуют весовым коэффициентам информационных элементов.

4. Формирование уточняющего множества M_k (множества контекстных слов) на основе контекстного анализа информационных элементов множества M_p.

5. Получение общего множества ключевых элементов, определяющего тематику текста: $M = M_p + M_k$.

Суть метода заключается в том, чтобы выделить первичный набор ключевых элементов за счет числа их повторений в тексте и затем дополнить его контекстом на основе анализа структурных особенностей матрицы.

Первые три пункта вряд ли нуждаются в дополнительных комментариях, рассмотрим подробнее порядок формирования множества M_k.

Одним из ключевых моментов является тот факт, что через информационный элемент поток может проходить множество раз. Тем более это справедливо для первичного множества ключевых элементов M_p, так как именно они выбраны из всего текста на том основании, что у них больше связей с другими информационными элементами. Окрестность информационного элемента — столбцы, а это множество информационных элементов и является контекстом. Анализ окрестности информационного элемента для выделения контекста данного элемента будем называть контекстным анализом, он может проводиться обратно ходу (предыдущий столбец) информационного потока ($-e$) или же по ходу (следующий столбец) информационного потока ($+e$).

Формирование множества M_k на основе контекстного анализа информационных элементов множества M_p выглядит следующим образом.

1. Выделяем в набор $A(w)$ множество всех потоков, проходящих через каждый информационный элемент $w \in M_p$, в некоторой окрестности заданной e. Объединяем все наборы $A(w)$ для каждого $w \in M_p$ в один общий набор $A(M_p)$ и обозначаем его A: $A = A(M_p)$.

В результате получим общий набор A, включающий в себя все потоки, проходящие через информационные элементы множества M_p, в некоторой окрестности заданной e.

2. Исключаем все информационные элементы из A, принадлежащие M_p: $A \to A'$.

3. Теперь из набора информационных элементов A' выделяем множество информационных элементов M_k: $A' \to M_k$.

При этом будем учитывать число повторяющихся информационных элементов и для каждого элемента множества M_k запишем число их повторений в наборе A': $M_k = \{k_1w_1, k_2w_2, ..., k_nw_n\}$, коэффициенты $k_1, k_2, ..., k_n$ перед w — это число повторений этих информационных элементов в наборе A'.

Все элементы множества M_k присутствуют в некоторой окрестности элементов множества M_p, и каждый информационный элемент $w \in M_p$ определяет центр некоторой окрестности (единица логической шкалы) в информационной структуре $D(W, C)$. Окрестность задается по информационным потокам, проходящим через w.

На рис. 2 приведен пример выделения окрестности информационного элемента w_5 некоторой информационной структуры (на рисунке приведен ее фрагмент) при $e = 1$. Для данного примера $M_k = \{w_9, w_3, w_6, w_1, w_2, w_8\}$, все коэффициенты равны 1.

Если рассматривать текст, то это означает, что для некоторого слова определяются все его вхождения в текст. Затем для каждого из вхождений определяются соседние с ним слова в пределах некоторого диапазона — окрестности столбца e (вперед и назад по тексту). Упорядоченные по

Рис. 2. Выделение словосочетаний (контекстов)

частоте встречаемости слова в тексте используются для определения весов этих слов в тематике.

Нужно отметить одну важную деталь, а именно, окрестности ключевых элементов из M_p могут пересекаться, если ключевые слова расположены в тексте рядом (в пределах e). Тогда при подсчете повторяющихся информационных элементов необходимо учитывать их индекс в информационном потоке и не считать повторно информационный элемент с одним и тем же индексом, чтобы веса элементов из множества M_k не превышали веса элементов множества M_p, так как мы определяем значимость данных элементов в тематике.

В итоге мы получаем общее множество ключевых элементов, определяющих тематику текста:

$$M = M_p + M_k;$$

$$M = \{k_1 w_1, k_2 w_2, ..., k_n w_n\}.$$

Весовые коэффициенты $k_1, k_2, ..., k_n$ определяют значимость того или иного информационного элемента в данной тематике. Эмпирический подбор коэффициента, определяющего границу, порог адекватности, возможен в рамках конкретной поисковой системы с учетом специфики анализируемых документов. Выбор конкретного порога или реализация оценки адекватности анализа того или иного текста на основе приведенного соотношения — это прерогатива разработчика поисковой системы [3].

Детальное рассмотрение этого вопроса требует дополнительной теоретической проработки и экспериментальных исследований, выходящих за рамки данной работы.

Заключение

Данный метод классификации тематики текста позволяет выделять тематику текста в виде множества ключевых слов с весами, характеризующими значимость данных слов в тематике. Достоинством метода является меньшая чувствительность к частотным выбросам частотно-значимых слов, так как выделение тематики не ограничивается одним лишь выделением этих слов. Преимущество метода — в дополнении первичного набора ключевых элементов (сформированных на основе частотно-значимых слов) контекстом, что позволяет точнее отобразить тематику текста.

Перспективным направлением исследований представляется поиск тематически близких документов на основе подобия их структуры, т. е. выявления общих структурных особенностей текстов, удовлетворяющих информационным потребностям пользователя. Сейчас выделение тематики предполагает анализ структуры текста и оформление результатов анализа в виде векторной модели для последующего поиска документов на основе этой модели.

Список литературы

1. **Chen Z.** Building Compressed Database Systems // Dissertation Presented to the Faculty of the Graduate School of Cornell University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy, Aug. 2002.

2. **Сулейманов А. Ш.** Структуризация научных документов и построения логико-семантической модели // Сб. тезисов докладов Международной научно-практической конференции "Компьютеры. Программы. Интернет. 2003". Киев/Украина. 2003. С. 125.

3. **Новиков А. И., Ярославцева Е. И.** Семантические расстояния в языке и тексте. М.: Наука, 1990.

УДК 004.912

А. Е. Ермаков, канд. техн. наук, рук. отдела компьютерной лингвистики ООО "ЭР СИ О", e-mail: ermakov@rco.ru

Извлечение знаний из текста и их обработка: состояние и перспективы

Выполнен анализ достижений в области компьютерной обработки знаний, содержащихся в текстах на естественном языке. Формулируются актуальные направления прикладных исследований, связанные с извлечением и обработкой знаний в текстах Интернета. Описывается экспериментальная система для оценки потребительских свойств товаров на основании анализа отзывов их потребителей, размещенных в социальной сети Интернет.

Ключевые слова: *извлечение знаний, обработка знаний, автоматизированные системы управления знаниями, компьютерный анализ текста, социальные сети Интернета.*

Знания в человеко-машинных системах

В настоящее время большое количество теоретических и практических исследований посвящено построению автоматизированных человеко-машинных систем, которые реализуют комплекс функций, обозначаемых словами "извлечение/управление/обработка знаний". В большинстве случаев под знаниями понимается нечто, выражающееся на естественном языке и изначально содержащееся либо в тексте, либо в голове человека-эксперта. Настоящая статья является результатом критического анализа практических достижений в этой области, попытки определения актуальных направлений развития и собственных экспериментальных исследований в выбранном направлении.

Обзор множества публикаций, некоторое число которых отмечено в ссылках [1—8], показывает, что в контексте рассматриваемой темы существуют два понимания термина "знание".

Первая точка зрения, акцентированная на прагматических аспектах и принятая в рамках направления *knowledge management*, представляет знания как данные, полученные в нужном месте и в нужное время для решения практической задачи, обычно для принятия решения, в том числе выполнения действия человеком или технической системой. При этом по своей структуре и способу хранения знания могут ничем не отличаться от прочих данных — любой фрагмент базы данных или полнотекстового архива документов превращается в знание, как только на него обращается взгляд заинтересованного потребителя. Именно положение этого взгляда — фокуса утилитарного интереса — определяет, какой фрагмент данных в настоящий момент интерпретируется как знание.

Вторая точка зрения, акцентированная на содержательных аспектах и принятая в рамках направления искусственного интеллекта, полагает, что знания отличаются от обычных данных прежде всего своей структурой. Именно к совокупности особым образом структурированных данных применимо понятие база знаний, подразумевающее:

- логическую упорядоченность данных на основании определенных критериев, устанавливаемых моделью предметной области (онтологией);
- представление данных в соответствии с определенной формальной моделью (семантической сетью, фреймами, набором продукций и др.);
- возможность получения новых данных из старых на основании определенного формального механизма;
- хранение данных в специальных структурах, обеспечивающих высокую эффективность типовых операций над ними (поиск на графах, анализ иерархий, логический вывод и др.).

Полный процесс управления знаниями (*knowledge management*) в общем случае содержит фазу их извлечения и фазу их обработки, которые реализуются в автоматизированной системе управления знаниями (АСУЗ). При этом технологические составляющие фазы извлечения определяют, каким образом данные превращаются в элементарные знания (аксиомы), а составляющие фазы обработки определяют то, как из элементарных знаний порождается новое знание, используемое для принятия решений. Два подхода к знанию нашли свое отражение в двух совершенно различных типах АСУЗ, автоматизирующих две различные фазы работы со знаниями, которые пока на практике не совмещаются.

Так, при прагматическом подходе к знаниям в центре внимания оказывается фаза их извлечения, поддержка которой обеспечивается информационно-поисковой составляющей АСУЗ. Фаза же обработки найденных первичных знаний реализуется за рамками АСУЗ, в голове аналитика или принимающего решения человека. В итоге, ключевым компонентом АСУЗ является поисковая машина, обеспечивающая оперативный отбор и доставку адекватной информации по запросам. Поисковый компонент либо является предметно-независимым, либо допускает простую настройку даже малоподготовленным пользователем, что определяет основное достоинство этого типа

АСУЗ: применимость к широкому кругу заранее неизвестных задач.

При содержательном подходе к знаниям фокус внимания направлен на фазу обработки элементарных знаний, получения из его кирпичиков нового знания на основании обобщения, сопоставления, логического вывода и т. п. В зависимости от формальной модели представления первичных знаний, на этой фазе могут применяться различные математические методы (статистический, регрессионный, кластерный, факторный анализ, анализ графов, вывод в системах продукций и др.). Результатом являются обобщения, выявление скрытых зависимостей и корреляций, прогнозы. В этом случае за рамками АСУЗ реализуется фаза извлечения первичных знаний, их формализации и размещения в базе знаний, логическая структура и фактическое наполнение которой зависят от особенностей предметной области и должны разрабатываться в тесном взаимодействии экспертов и инженера по знаниям. Узкая специализация базы знаний и трудоемкость ее разработки являются недостатками АСУЗ, воплощающих содержательный подход. Достоинством же АСУЗ является возможность быстрого получения решения для тех типовых задач, на решение которых они ориентированы.

Прикладные АСУЗ сегодня

Основными потребителями знаний сегодня выступают следующие группы людей:

1) руководящие работники, принимающие управленческие решения;

2) аналитики, составляющие обзоры, прогнозы и рекомендации для группы 1, в том числе сотрудники спецслужб;

3) узкие сообщества профессионалов в некоторых областях, для которых разрабатываются специализированные системы — экспертные системы в медицине, геологии; системы извлечения формул органических соединений из научных публикаций в химии и т. п.;

4) прочие работники научной и информационной сферы, нуждающиеся в своевременном и полном получении информации для производства интеллектуальной продукции (например, структурированных новостей по интересующим разделам науки и техники, общественно-политической жизни);

5) непрофессиональные потребители — люди, желающие использовать знания для бытовых нужд в целях принятия решения, например, о выборе модели товара при покупке, выборе поставщика услуг или способа действия в определенной ситуации (юридические и медицинские вопросы, устранение неисправностей техники).

Как показывают результаты обзора литературы, в том числе интернет-материалов, представленных фирмами-производителями программных решений в области *knowledge management*, на сегодняшний день все внимание теоретиков и практиков направлено на удовлетворение потребностей групп 1 и 2, что, по-видимому, связано с наибольшей ожидаемой здесь отдачей от капиталовложений. Так, в автоматизированных информационных системах, позиционируемых на рынке как системы поддержки принятия решений, конкурентной разведки, *business intelligence*, уже присутствует множество подсистем, реализующих те или иные функции извлечения, накопления, поиска и генерации новых знаний. Также определенное внимание привлекают к себе сообщества группы 3, для которых, обычно за счет бюджетного финансирования, разрабатываются специализированные прикладные информационные системы, включающие в себя средства *data mining* и *text mining*, экспертные системы.

Группа 4 и особенно группа 5, к которой относится каждый человек, обделены вниманием разработчиков и, несмотря на свободный доступ к потенциальному источнику знаний — Интернету, ограничены в инструментарии извлечения знаний простейшими (с точки зрения потребительских функций) поисковыми машинами типа Google или Яндекс. Отчасти это мотивировано необозримой широтой интересов данных групп, отсутствием ограниченной предметной области.

Окончательно, автору не удалось обнаружить не только полноценной АСУЗ, совмещающей в себе фазу извлечения знаний из текста с фазой их обработки, но даже убедительного примера практически полезной работы такой системы. Прикладных программ, использующих методы искусственного интеллекта, способных нетривиально перерабатывать извлеченные из текста элементы знаний (интерпретировать, обобщать, выявлять зависимости, прогнозировать и т. п.), сегодня не существует даже для английского языка. Такая ситуация обусловлена, по-видимому, двумя причинами. Во-первых, слабым распространением систем лингвистического анализа текста, способных интерпретировать отношения между словами и потому действительно извлекать знания как некие элементы, обладающие внутренней структурой и пригодные для нетривиальной смысловой обработки искусственным мозгом, — такие системы понимания текста на мировом и российском рынках только недавно начали появляться и еще не успели обрасти приложениями: Net Owl (www.netowl.com), Attensity (www.attensity.com), RCO Fact Extractor (www.rco.ru). Во-вторых, потенциально низкой достоверностью автоматически извлекаемых из текста утверждений и фактов,

что обусловлено как несовершенством алгоритмов интерпретации текста, так и низким качеством источников информации, поскольку практически интересно извлечение знаний не из научной литературы, а из различного рода текстовых "помоек", к каковым относятся социальные сети Интернет, современные СМИ, и даже архивы научно-технических отчетов.

В итоге, несмотря на бум вокруг необходимости извлечения знаний из текста, их переработки и утилизации, поднятый сегодня разработчиками и продавцами АСУЗ, создается впечатление, что на практике такие системы пока бесполезны, во всяком случае, за пределами узко специализированных областей, что, однако, неверно.

Интернет как источник знаний

Современное состояние Интернета позволяет рассматривать его в качестве источника самых разнообразных знаний, которые скрываются в корпоративных интернет-порталах и домашних страничках экспертов, блогах и форумах, аналитических статьях. Сегодня в сети существуют тысячи профессиональных сообществ, объединенных интересами в общей сфере: обсуждение научных проблем и технологий (нанотехнологии, искусственный интеллект), потребление определенных классов товаров и услуг (автомобили, туристическое обслуживание), общественно-политические события и исторические проблемы (межнациональные отношения, история церковного раскола).

Из разбросанных по Интернету знаний стоит выделить следующие классы, представляющие утилитарный интерес для обширных целевых аудиторий:

- знания о технических и качественных характеристиках товаров и услуг, позволяющие провести их сравнение и выбрать оптимальный вариант для покупки — электронные устройства и бытовая техника, автомобили; услуги по туризму, ремонту, лечению и т. д.;
- знания о способах и особенностях использования технологий — ремонт и отделка жилья, устранение неполадок автомобилей и т. п.;
- научные, технологические и общественно-политические события — открытия и находки, появление новых продуктов и технологий, происшествия и прогнозы;
- полезные факты различной природы, характеризующие деятельность людей и организаций — историко-биографические факты, взаимоотношения и связи.

Как особый источник информации сегодня следует выделить социальные сети Интернета (блоги, форумы, конференции и прочие виды электронных сообществ), терабайты текстовых сообщений в которых содержат реальные элементы утилитарных знаний, полученные людьми в результате их профессиональной и бытовой деятельности.

В качестве примера рассмотрим известный в Интернете блог "Живой Журнал" (http://www.livejournal.ru/) — сеть электронных дневников пользователей, которые делают записи (посты) в своих дневниках и комментарии на записи других пользователей в своих и чужих дневниках. По нашим оценкам, летом 2007 года русскоязычная часть блога содержала более 75 тыс. тематических сообществ, записи в блоге оставили более 1 млн 200 тыс. пользователей, а в день в среднем добавлялось около 100 тыс. постов и 400 тыс. комментариев. Следует ожидать, что объем информации и число пользователей будут увеличиваться.

К актуальным задачам, которые сегодня должны и могут решаться АСУЗ в Интернете, по мнению автора, относятся следующие:

- поиск и извлечение элементов знания, явно присутствующих в текстах в виде: а) утверждений (*лекарство Антипилин — полная ерунда; вероятная причина свиста под капотом автомобиля в сырую погоду — слабое натяжение ремня генератора*); б) фактов (*после принятия Антипилина может подниматься давление; фирма Пежо отозвала 20 000 автомобилей из-за возможного возгорания в системе электроусилителя руля*);
- порождение обобщенного знания, скрытого в совокупности частных утверждений и/или фактов, например, порождение выводов типа *препарат Антипилин имеет меньше побочных эффектов, чем Глипирон* (на основании анализа отзывов больных) или *типичная причина поломок автомобиля Форд Фокус — засорение бензонасоса* (на основании анализа отзывов владельцев автомобилей).

Возможность практического решения подобных задач сегодня исследуется в компании "ЭР СИ О", с использованием собственного лингвистического анализатора русского текста (http://www.rco.ru).

Опыт извлечения знаний из Интернета: оценка потребительских свойств товаров

Задача разработанной экспериментальной АСУЗ состояла в том, чтобы для каждой модели автомобиля "выловить" положительные и отрицательные отзывы и классифицировать их: "за что хвалят/ругают?".

Для извлечения знаний было выбрано крупнейшее из нескольких десятков автомобильных сообществ "Живого журнала" сообщество AUTO_RU: "Все об автомобилях" (http://community.livejournal.com/avto_ru/). За 2007 год сообщество содержит записи 3 тыс. авторов постов и

6 тыс. авторов комментариев, всего около 500 тыс. сообщений, порожденных 19 тыс. постов, с объемом русскоязычного текста около 60 Мбайт.

Web-интерфейс пользователя АСУЗ, обеспечивающий просмотр извлеченных из текста знаний, построен на базе трех взаимосвязанных окон (рис. 1).

Окно в левой части экрана содержит перечень марок автомобилей, по которым в базе содержатся выделенные знания. Тривиальная операция отождествления синонимичных названий (*БМВ = бумер = бэха, ВАЗ 2109 = Девятка*) и их группировки по производителям (*ВАЗ = {ВАЗ 2105, ..., жигули, жига, таз, тазик, ...}*) пока не проводилась. Вход в базу от выбранной марки или модели позволяет просмотреть все извлеченные знания, относящиеся к ней, которые сгруппированы по типу: *общая характеристика: хорошо; комфорт: хорошо; надежность: плохо* и т. п. На каждый тип знаний указано число записей, содержащихся в базе.

Окно в верхней части экрана справа содержит таблицу с записями, относящимися к выбранному типу знания по выбранной марке. Первый столбец содержит название марки; второй столбец — тип извлеченного знания + конкретное ключевое слово, извлеченное из текста, например, *безопасность: хорошо: УСТОЙЧИВЫЙ*; третий столбец — дополнительную извлеченную информацию (дату, место), а четвертый столбец — характеристики достоверности: число упоминаний знания в текстах и характеристики наиболее достоверного из источников, из которого были извлечены знания: его относительный вес в сообществе (индекс цитирования) и характер (средняя позитивность или негативность высказываемых им оценок).

Окно в нижней части экрана справа отображает цитаты из источника, в которых упоминается извлеченное знание — предложения исходного текста, выделенные анализатором текста. Для каждой цитаты указывается источник с его характеристиками, а также ссылка на полный текст сообщения для просмотра контекста цитаты.

Для оценки высказываний об автомобилях с точки зрения характеристик их потребительских свойств

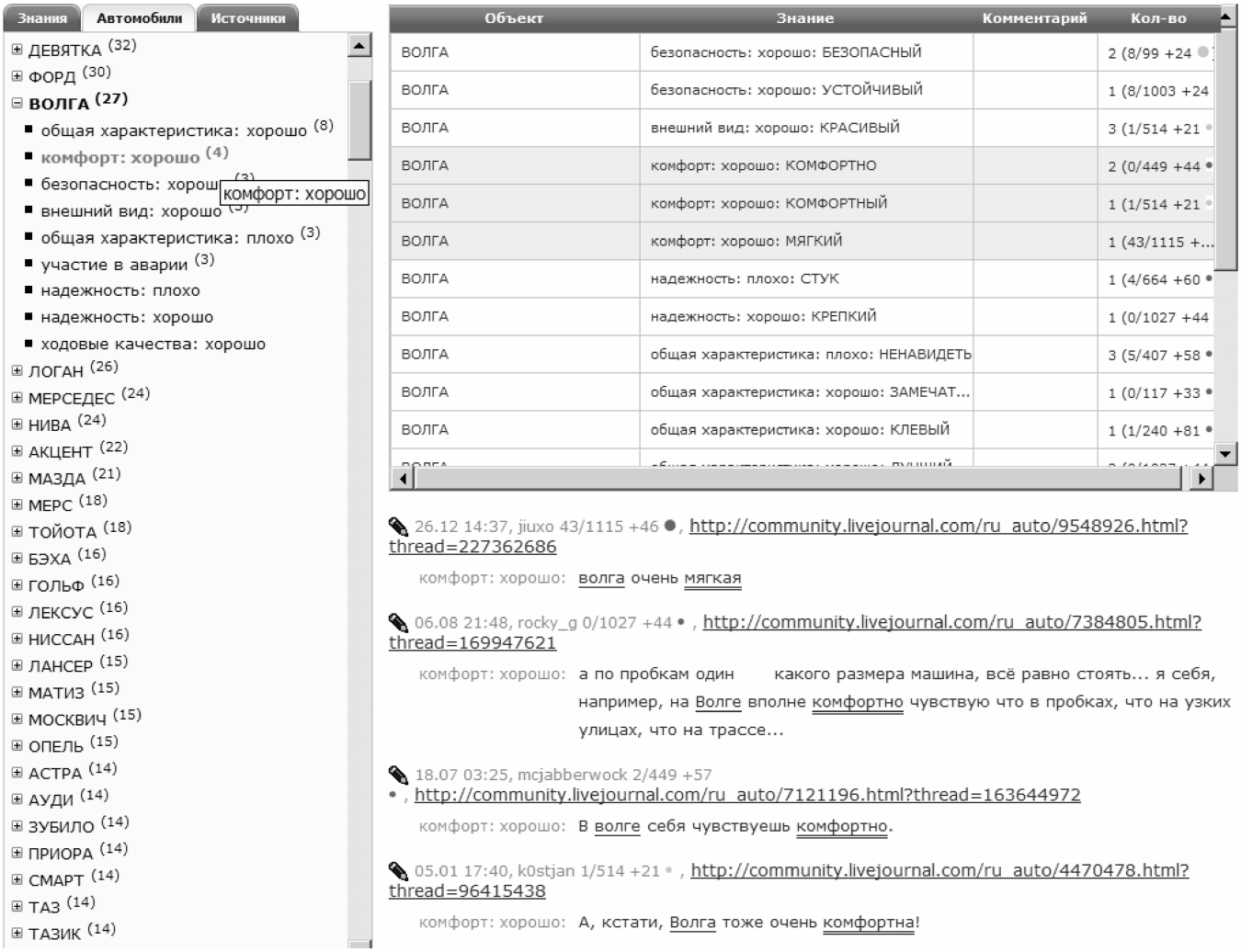


Рис. 1. Вход в базу знаний от выбранного объекта (*Волга*) с просмотром знаний выбранного типа (*комфорт*: *хорошо*)

{*положительная/отрицательная*) была разработана экспериментальная онтология, содержащая:

а) более 700 различных наименований марок автомобилей (*Легенда*, *Сивик*, *Приора*) и фирм-производителей (*BMW*, *БМВ*, *ВАЗ*). Первоначальный список был получен с сайта http://www.auto.ru/, после чего все наименования были расширены синонимами с учетом вариантов написания и известных "народных" названий (*Mitsubishi = Мицубиси = Митсубиши = Мицу; BMW = БМВ = бумер = бэха*). Конкретные символьно-цифровые обозначения моделей, упоминаемые в тексте (*BMW 325i, ВАЗ 21053*), не включались в словарь, а распознавались по формальным правилам;

б) более 1200 терминов в 24 группах, среди которых:

- 211 наименований узлов автомобиля (*движок, коробка передач, ходовая часть*);
- 71 наименование свойств, которые классифицированы на восемь оцениваемых групп (*ходовые качества, комфорт, безопасность, надежность, ...*);
- 882 наименования оценок характеристик узлов и свойств, включающие прилагательные, существительные, глаголы и наречия (*крутой, поломка, глючить, отстойно*), в том числе и нелигитимную лексику;
- 37 эмоциональных характеристик (*любить, жалоба, плеваться*);

в) около 100 семантических шаблонов, описывающих возможные синтаксические связи в предложении между 24 группами терминов (б) из онтологии. На рис. 2 приведен пример одного из шаблонов, предназначенного для семантической интерпретации фраз, построенных по схеме типа: *Размер багажника на Outlander XL вызывает восторг*; *Вид салона Нексии приводит в бешенство*.

Описание семантического интерпретатора с соответствующим лингвистическим анализатором текста приведено в работе [9].

Шаблон задает лексико-грамматические ограничения на искомую конфигурацию связей между словами в тексте, которые определяются синтаксическим анализатором. В вершинах (прямоугольниках со скругленными краями) указываются ограничения на конкретные слова (*Name = = "ВЫЗЫВАТЬ"*), части речи (*SpeechPart = "Noun"* — существительное) или семантические разряды слов (*SemanticType = "Artifact : CarUnit"* — узел автомобиля). В эллипсах описываются ограничения на синтактико-семантические связи между словами: тип связи (*RelationName = = "принадлежность"*), предлог (*RelationConnector = = "НА"*), семантический падеж {*RelationCase = = "И"* — именительный, субъект). Окончательно, такой шаблон параметризуется множеством конкретных слов из онтологии — названиями эмоций типа *восторг*, *бешенство* параметризуется узел с ограничениями *SpeechPart* = "*Noun*", названиями узлов автомобиля (*багажник, салон*) и их характеристик (*размер, вид*) параметризуются узлы с ограничениями *SemanticType = "Artifact : CarUnit*" и *SemanticType = "Artifact : CarCharacteristic".*

Автоматизированная разработка онтологии проводилась на базе анализа языкового материала сообщества AUTO_RU "Живого журнала" с помощью средств компьютерного анализа текста [10], на что было затрачено около 5 чел./дней работы эксперта-автомобилиста (формирование словаря терминов, их классификация по разделам онтологии) и около 50 чел./дней работы лингвиста (отбор и систематизация типовых способов выражения,

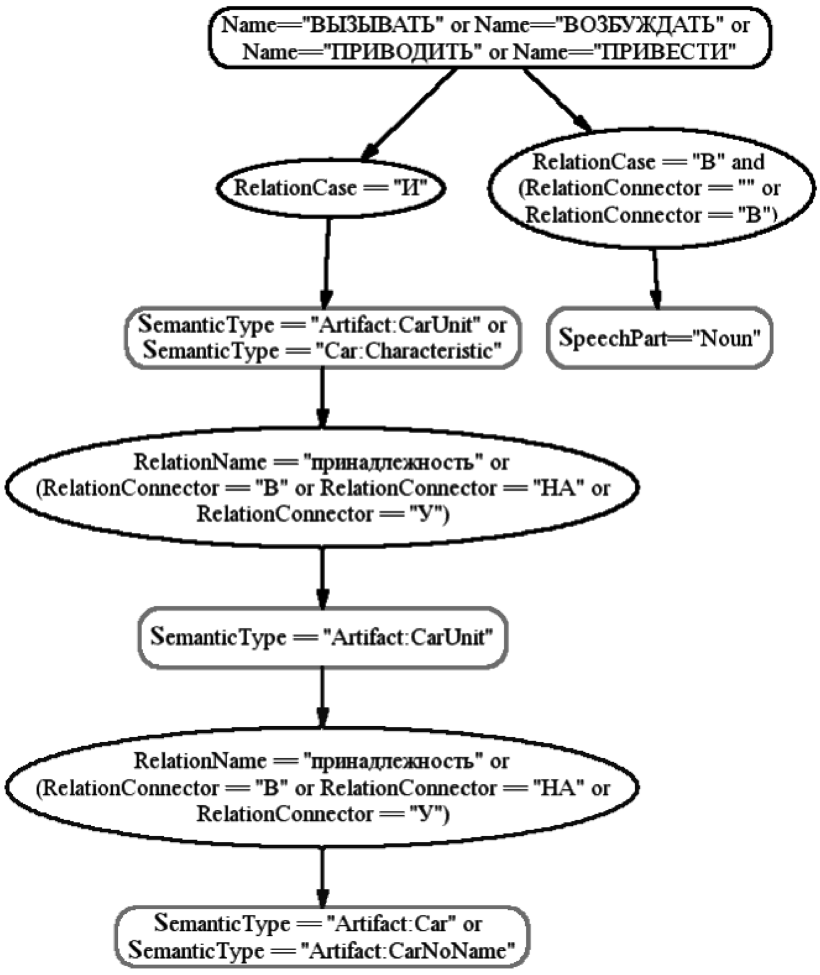


Рис. 2. Семантический шаблон для извлечения оценки автомобиля, которая выражается существительным в конструкциях вида: *Размер багажника на Outlander XL вызывает восторг*; *Вид салона Нексии приводит в бешенство*

создание соответствующих им семантических шаблонов, составление соответствующих словарей оценочных слов, общая настройка и тестирование созданного лингвистического обеспечения).

В итоге, из 500 тыс. сообщений "Живого Журнала" (60 Мбайт текста) было извлечено всего более 5 тыс. оценок автомобилей, их узлов и характеристик, из которых более 1 тыс. (795 хороших и 328 плохих) оценок привязано к маркам автомобилей, а более 4 тыс. оценок узлов и характеристик программе не удалось привязать к конкретным маркам (в предложениях с синтаксически невыраженным референтом, наподобие: *А движок — просто зверь*). В результате достигнута точность 84 %, а полнота извлечения около 20 %.

Анализ ошибок показывает, что как точность, так и полнота могут быть еще повышены за счет дальнейшей доработки онтологии, но незначительно. В частности, принципиальную проблему представляет интерпретация фраз, содержащих отрицание, выраженное сложным образом, например:

- логически: *Я нигде, никогда, не писал, что Мерседес, БМВ и Ягуар имеют высокую надежность*; *Расскажите про слабую подвеску CR-V кому-то другому*; *А вообще чтоб в такую погоду на 60 Лексус занесло — это бред*!
- метафорически: *За 7 лет от надежности Сивика остались одни воспоминания*; *Вольво с40 за милую душу хлебают водички и радуют владельца счетом на 2000 дензнаков ненашенских.*

Заключение

Автоматизированные системы извлечения и обработки знаний, не нашедшие пока практического применения за пределами узкоспециализированных областей, имеют реальную перспективу войти в повседневную жизнь в ближайшем будущем, используя социальные сети Интернета в качестве источника знаний.

В проведенном эксперименте удалось показать практическую возможность извлечения из социальных сетей Интернета утилитарных знаний, полезных каждому человеку. Учитывая, помимо большого количества грамматических и орфографических ошибок, особый аграмматичный и аорфографичный стиль этого "жанра" текстов (http://ra.wikipedia.org/wiki/)Жаргон_падонков), большое количество сленговой лексики, следует признать результаты эксперимента более чем удовлетворительными. Окончательно ожидаемая точность в районе 90% вселяет надежду на возможность извлечения знаний из "интернет-помоек" с приемлемым качеством, поскольку недостаточная полнота (20 %) легко компенсируется избытком информации. Следует также учесть, что задача решалась в предельно сложной постановке — автоматически определить "хвалят или ругают?". Подобная задача уже решалась автором для текстов СМИ [11]. И там и здесь, похоже, оказывается не столь важно понять — хвалят или ругают, важнее понять — за что? В более узкой постановке — поиск оценочных высказываний, их отнесение к оцениваемым объектам (что хвалят/ругают?) и классификация (за что? — за двигатель, подвеску, проходимость, разгон, надежность и т. п.) — задача решается с точностью, близкой к 100 %.

Область дальнейших исследований, определяющая окончательную эффективность утилизации извлекаемых знаний, — это разработка концепции пользовательского интерфейса АСУЗ, позволяющего минимизировать время на изучение отзывов по интересующим объектам и выработку их сравнительной оценки. В этой области могут помочь методы автоматической оценки достоверности извлеченных знаний, оценки компетентности их источников (авторов сообщений) и степени доверия к ним. Поиск экспертов, мнению которых можно доверять — это один из путей к скорейшему принятию решения.

На взгляд автора, решение рассмотренных задач имеет приоритетное народнохозяйственное значение в сфере мирных приложений технологий обработки знаний, которые могут быть внедрены в массовые интеллектуальные системы поддержки принятия решений, в частности, помогающие в выборе товаров и услуг на основании анализа отзывов их потребителей.

Список литературы

1. **Гаврилова Т., Хорошевский В.** Базы знаний интеллектуальных систем: Учеб. для вузов. СПб.: Питер, 2000. 384 с.
2. **Девятков В. В.** Системы искусственного интеллекта: Учеб. пособие для вузов. М.: Изд-во МГТУ им. Н. Э. Баумана, 2001. 352 с.
3. **Букович У., Уильямс Р.** Управление знаниями: руководство к действию: Пер. с англ. М.: ИНФРА-М, 2002. 504 с.
4. **Джанетто К., Уиллер Э.** Управление знаниями: Руководство по разработке и внедрению корпоративной системы управления знаниями: Пер. с англ. М.: Добрая книга, 2005. 192 с.
5. **Колесов А.** А управлять — так знаниями! // Byte. 2002. № 2.
6. **Петелин Д.** Свалки данных и системы управления знаниями // PC Week (RE). 2006. № 19.
7. **Ландэ Д. В.** Поиск знаний в Internet. М.: Диалектика-Вильямс, 2005. 272 с.
8. **Ашманов И.** Информация и знания: невидимая грань. URL: http://newasp.omskreg.ru/intellect/f5.htm.
9. **Киселев С. Л., Ермаков А. Е., Плешко В. В.** Поиск фактов в тексте естественного языка на основе сетевых описаний // Компьютерная лингвистика и интеллектуальные технологии: Тр. Междун. конф. Диалог'2004. М.: Наука, 2004.
10. **Ермаков А. Е.** Автоматизация онтологического инжиниринга в системах извлечения знаний из текста // Компьютерная лингвистика и интеллектуальные технологии: Материалы ежегодной Международной конференции "Диалог", Бекасово, 4—8 июня 2008 г. Вып. 7 (14). М.: Изд-во РГГУ, 2008.
11. **Ермаков А. Е., Киселев С. Л.** Лингвистическая модель для компьютерного анализа тональности публикаций СМИ // Компьютерная лингвистика и интеллектуальные технологии: Труды Международной конференции Диалог'2005. М.: Наука, 2005.

УДК 025.4

В. А. Васенин, д-р физ.-мат. наук, зав. лаб.,
С. А. Афонин, канд. физ.-мат. наук, вед. науч. сотр.,
А. С. Козицын, канд. физ.-мат. наук, вед. науч. сотр.,
НИИ механики МГУ им. М. В. Ломоносова,
e-mail: vasenin@msu.ru

Автоматизированный анализ текстовой информации

Рассматривается система, предназначенная для поиска информации и тематической фильтрации входящего информационного потока и ориентированная, в первую очередь, на использование в сети Интернет либо в крупной корпоративной информационной системе.

Ключевые слова: *коллекция документов, автоматическая обработка, значимость, информация, поисковые системы, индексация текстов.*

Быстрое развитие средств хранения, обработки и передачи данных в течение последней четверти века повлекло за собой еще более быстрое увеличение объемов разноплановой информации, представленной в метасети Интернет. От способов ее извлечения, систематизации и оперативного анализа сегодня зависит успех деятельности не только в науке, технике и в образовании, но и в области материального производства, в социально-экономической и политической сферах.

Эффективность деятельности многих хозяйствующих субъектов в настоящее время определяется своевременностью, полнотой и качеством получаемой ими информации, а также возможностями по ее обработке под свои потребности. Только в верхнем, которое принято именовать статическим, информационном поле метасети в 2003 году размещалось около 8 млрд документов, на настоящее время около 50 млрд, и это количество удваивается каждые 8—12 месяцев. Объем динамично меняющегося информационного пространства (базы данных и аналогичные источники), согласно результатам последних исследований, оценивается от 10 до 100 Пбайт (10^{16}—10^{17} байт). В связи с этим существующих и традиционно используемых методик поиска информации оказывается недостаточно. Появляется необходимость в разработке поисковых систем нового поколения, которые используют методы интеллектуального анализа текстов и позволяют пользователю наиболее точно описывать свою информационную потребность с тем, чтобы добиться наибольшей релевантности и полноты поиска [1].

В современных условиях для эффективной работы необходимы системы, которые в автоматизированном режиме их эксплуатации способны осуществлять комплексный интеллектуальный анализ больших объемов текстовой информации с компактным визуальным представлением его результатов. Чрезвычайно востребованы подобные системы в крупных корпоративных структурах, эффективное управление которыми напрямую зависит от большого числа динамично меняющихся факторов, а также в других предприятиях и организациях государственного сектора экономики, структурах среднего и малого бизнеса.

Сформулируем основные задачи, которые могут быть решены с помощью рассматриваемой далее автоматизированной системы тематического анализа (АСТАИ). К основным, предметно независимым задачам, которые призвана решать разрабатываемая АСТАИ, следует отнести:

- тематический поиск информации;
- тематическую фильтрацию входящего информационного потока;
- поиск с использованием структуры документов.

Особенностью рассматриваемой системы является использование алгоритмов автоматического тематического анализа текстов как при первичной загрузке документов, так и в ходе реализации поиска. В системе имеется набор базовых классификаторов, которые настраиваются экспертами на этапе обучения системы. Авторизованным пользователям системы предоставляется возможность создания и использования собственных классификаторов, уточняющих базовые. Еще одной особенностью системы является использование алгоритмов автоматического выделения имен и названий объектов, а также составление их кратких описаний.

Отметим тот факт, что коммерческие аналоги подобных систем, как правило, являются закрытыми, они поставляются без открытых исходных кодов. В связи с этим ограничены возможности их модификации под постоянно и объективно изменяющиеся потребности пользователей. Кроме того, отсутствие открытого исходного кода ограничивает возможности верификации и сертификации подобных систем. Как следствие, уменьшаются оценочные уровни доверия к ним в соответствии с нормативными документами, регламентирующими деятельность по обеспечению безопасности используемых в их составе информационных технологий.

В основе архитектурных решений АСТАИ положена концепция построения распределенных систем на базе сервисно-ориентированной архитектуры. Основные функциональные элементы системы выполнены как web-сервисы с описанным интерфейсом. Ориентация на открытые протоколы взаимодействия предоставляет возможность тесной интеграции созданной системы с су-

ществующими программными средствами сторонних производителей. Указанный подход позволяет также быстро наращивать функциональные возможности АСТАИ и проводить апробацию новых решений в области интеллектуального анализа текстов.

Полностью текст, отражающий состояние работ по АСТАИ, опубликован в **Приложении к журналу "Информационные технологии" № 4 за 2009 год.** В нем представлены результаты проведенных авторами поисковых исследований, опытно-конструкторских работ и решения прикладных задач, направленные на изучение и развитие методов и средств интеллектуального анализа текстов, на разработку архитектуры программного комплекса для поиска данных в больших коллекциях текстовых документов. В Приложении подробно рассматриваются некоторые аспекты практической реализации новых формальных моделей, архитектурных и технологических решений, которые могут использоваться в специализированных поисковых информационных системах для интеллектуальной обработки текстовых данных. Приводится описание разработанного на их основе прототипа автоматизированной системы для тематического анализа информации, которая является второй версией разработанной ранее автоматизированной системы информационного обеспечения [2].

Работа выполнена при поддержке гранта РФФИ № 09-07-00366-а.

Список литературы

1. **Васенин В. А., Афонин С. А., Козицын А. С., Шундеев А. С.** Поиск в сверхбольших хранилищах данных и высокопроизводительные системы с массовым параллелизмом // Программные системы: теория и приложения. Тр. междунар. конф. г. Переславль-Залесский, май 2004 г. 2004. Т. 1. С. 211—228.

2. **Козицын А. С.** Система тематического мониторинга распределенной текстовой информации // Матер. Третьей междунар. конф. по проблемам управления. 2006. С. 757—761.

КОМПЬЮТЕРНАЯ ГРАФИКА

УДК 532.5+536.24

К. В. Максименко-Шейко*, канд. физ.-мат. наук,
А. В. Толок,** д-р техн. наук, проф.,
Т. И. Шейко*, д-р техн. наук, проф.,
* ИПМаш НАН Украины,
ХНУ им. В. Н. Каразина, г. Харьков,
** ИПУ РАН им. В. А. Трапезникова, г. Москва

R-функции и аналитическое описание геометрических объектов, обладающих симметрией

Предложены новые подходы, раскрывающие возможности применения теории R-функций к построению нормализованных уравнений сложных обладающих симметрией геометрических объектов из различных прикладных областей.

Ключевые слова: *геометрический объект, симметрия, трансляционный элемент, преобразование координат, R-функции.*

Введение

К настоящему времени опубликовано большое число работ (например, [1—6]), посвященных методу R-функций и его применениям. Ключевым в использовании этого метода является предоставляемая им возможность построения для сложных геометрических объектов (ГО) Ω нормализованных уравнений вида $\omega(x, y, z) = 0$, где $\{\omega(x, y, z)\}$ — элементарные функции, представленные единым аналитическим выражением. Этим самым была решена поставленная еще во времена Декарта обратная задача аналитической геометрии. При этом существенно важно, насколько простыми с вычислительной точки зрения оказываются формулы $\omega(x, y, z)$. Интересными в этом аспекте являются ГО, обладающие симметрией.

Феномен симметрии, наблюдаемый в природе, начиная с молекул, кристаллов, растительного и животного мира, был позаимствован и перенесен человеком в архитектуру, строительство, машиностроение не только из соображений эстетики, но и для достижения рациональности, равновесности, устойчивости, надежности, облегченности и удешевления зданий, сооружений, машин, механизмов и других конструкций и изделий. Все это нашло отражение в геометрии, алгебре, анализе, других разделах математики и стимулировало изучение законов симметрии, методов их описания и применения, постановку новых проблем. Так возникла проблема построения уравнений сложных ГО, состоящих из элементов, многократно повторяющихся по тем или иным законам симметрии.

Суть этой проблемы заключается в том, что для задания геометрического объекта, в формировании которого многократно тиражируется один и тот же ГО, достаточно располагать лишь уравнением этого трансляционного элемента и информацией о типе симметрии. Это означает, что для построения уравнения ГО, обладающего симметрией, вовсе необязательно писать уравнения всех элементов рассматриваемого объекта, а затем с помощью R-функций осуществлять их сборку. Возможен другой путь: располагая уравнением трансляционного элемента, выполнить в нем такое преобразование координат, которое позволило бы получить единое уравнение без применения R-операций, требующих значительных вычислительных затрат. Первые попытки в этом направлении были предприняты в работах [1, 2]. Однако многие типы симметрии не были затронуты.

Целью данной работы является совершенствование конструктивных средств теории R-функций, основанных на преобразованиях координат, для осуществления построения нормализованных уравнений ГО с последующей их трансляцией.

Основная часть

В работах [3—5] был предложен целый ряд преобразований координат, позволяющих транслировать ГО по прямой, по отрезку прямой и по окружности. Этот материал явился основой для дальнейшего развития методики и выяснения дополнительных возможностей при ее использовании.

Преобразование координат для построения уравнений, соответствующих ГО с симметрией трансляции (бесконечного тиражирования) вдоль прямой. Пусть $\Sigma_0 = [\sigma_0(x, y \geqslant 0)]$, $\sigma_0 \in C^m(\Omega)$, есть область, симметричная относительно оси ординат, которая может быть заключена в вертикальную полосу $-a < x < a$, а $\Sigma_i = [\sigma_0(x - hi, y) \geqslant 0]$, $i = 0, \pm 1, \pm 2, \ldots$, — области, полученные смещением Σ_0 вдоль оси абсцисс на значения, кратные $h > 2a$, где h — шаг трансляции. Тогда граница $\partial\Omega$ области $\Omega = \bigcup\limits_{i=-\infty}^{i=\infty} \Sigma_i$ может быть задана уравнением

$$\omega(x, y) \equiv \sigma_0(\mu(x, h), y) = 0, \qquad (1)$$

где $\mu(x, h) = \dfrac{4h}{\pi^2} \sum\limits_{i=1}^{\infty} \dfrac{(-1)^{i+1}}{(2i-1)^2} \sin\dfrac{(2i-1)x\pi}{h}$.

Если же область Σ_0 несимметрична относительно оси Oy, то уравнение границы области Ω может быть написано в виде

$$\omega(x, y) \equiv \left\{\sigma_0(\mu(x, h), y) \wedge_\alpha \cos\frac{x\pi}{h}\right\} \vee_\alpha$$

$$\vee_\alpha \left\{\sigma_0(\mu(x - h, h), y) \wedge_\alpha \cos\frac{(x-h)\pi}{h}\right\} = 0. \qquad (2)$$

В первой фигурной скобке стоит выражение, нули которого образуют границу транслируемой области без зеркальных элементов, а во второй фигурной скобке — нули той же области, но смещенной вдоль оси абсцисс на величину h и занявшей место ее зеркального образа. Соединение этих областей и дает требуемую область.

Однако описанный подход построения уравнений чертежей, обладающих симметрией трансляции, не всегда возможен. А именно, если соседние области Σ_i не могут быть отделены вертикальными прямыми, т. е. $h \leqslant 2a$ (шаг трансляции не превосходит ширины ГО Σ_0), но можно отделить такими прямыми области Σ_i, Σ_{i+2}, то функция $\omega(x, y)$ может быть представлена в виде

$$\omega(x, y) \equiv \left\{\left[\sigma_0(\mu(x, 2h), y) \wedge_\alpha \cos\frac{x\pi}{2h}\right] \vee_\alpha\right.$$

$$\left.\vee_\alpha \left[\sigma_0(\mu(x - 2h, 2h), y) \wedge_\alpha \cos\frac{(x-2h)\pi}{2h}\right]\right\} \vee_\alpha$$

$$\vee_\alpha \left\{\left[\sigma_0(\mu(x - h, 2h), y) \wedge_\alpha \cos\frac{(x-h)\pi}{2h}\right] \vee_\alpha\right.$$

$$\left.\vee_\alpha \left[\sigma_0(\mu(x - 3h, 2h), y) \wedge_\alpha \cos\frac{(x-3h)\pi}{2h}\right]\right\} = 0. \qquad (3)$$

Используя преобразование поворота осей координат, легко получить аналогичное уравнение для произвольной прямой.

Преобразование координат для построения уравнений, соответствующих ГО с точечной симметрией циклического типа. Пусть $\Sigma_0 = [\sigma_0(x, y) \geqslant 0]$ есть некоторая область. Тогда $\Sigma_1 = [\sigma_1'(x, y) \equiv \sigma_0(x - r_0, y) \geqslant 0]$ есть результат смещения этой области на расстояние r_0 вдоль оси абсцисс. Поставим задачу: написать уравнение $\omega(x, y) = 0$ ГО $\partial\Omega$, представляющего собой границу области $\Omega = \bigcup\limits_{k=1}^{k=n} \Sigma_k$, являющейся соединением n областей Σ_i ($i = 1, \ldots, n$),

полученных в результате поворота области Σ_1 вокруг начала координат на углы $\frac{2\pi k}{n}$ $(k = 0, 1, ..., n-1)$.

Если указанная выше функция $\sigma_0(x, y) \in C^{(m)}$ нормализована, а область $\Sigma_0 = [\sigma_0(x - r_0, y) \geqslant 0]$ симметрична относительно оси абсцисс и может быть размещена внутри сектора $-\alpha \leqslant \theta \leqslant \alpha$, $0 < \alpha < \frac{\pi}{n}$, то нормализованное уравнение $\partial\Omega = [\omega(x, y) = 0]$, $\omega \in C^{(m)}$, может быть написано в виде

$$\omega(x, y) \equiv \sigma_0(r\cos\mu(n\theta) - r_0, r\sin\mu(n\theta)) = 0, \quad \left(r = \sqrt{x^2 + y^2}, \theta = \operatorname{arctg}\frac{y}{x}\right), \tag{4}$$

где $\mu(n\theta) = \frac{8}{n\pi}\sum_k (-1)^{k+1}\frac{\sin\left[(2k-1)\frac{n\theta}{2}\right]}{(2k-1)^2}$.

Если область Σ_0 несимметрична относительно оси абсцисс, то функция $\omega(x, y)$, нормализованная на границе области $\Omega = [\omega(x, y) \geqslant 0]$, может быть представлена формулой

$$\omega(x, y) \equiv \left[\sigma_0(r\cos\mu(n\theta) - r_0, r\sin\mu(n\theta)) \wedge_\alpha \cos\left(\frac{n\theta}{2}\right)\right] \vee_\alpha \vee_\alpha \left[\sigma_0\left(r\cos\mu\left(n\left(\theta - \frac{2\pi}{n}\right)\right) - r_0, r\sin\mu\left(n\left(\theta - \frac{2\pi}{n}\right)\right)\right) \wedge_\alpha \cos\left(\frac{n}{2}\left(\theta - \frac{2\pi}{n}\right)\right)\right] = 0. \tag{5}$$

В работах [3—5] приведены многочисленные примеры применения преобразований (1)—(5). Для иллюстрации возможностей предложенного метода, позволяющего максимально упростить процесс построения уравнений симметричных ГО, рассмотрим случай, когда правильные n_o-угольники транслируются в правильном n_b-угольнике n_d раз (рис. 1, см. третью сторону обложки). При этом все построения выполним с помощью двух прямых: $x - R_1 \geqslant 0$ и $R_v - x \geqslant 0$ (рис. 2, *а*).

Для построения уравнения правильного n_o-угольника воспользуемся уравнением прямой $x - R_1 \geqslant 0$ и формулой (4), а затем, перенося начало координат в точку $x = R_d$, оттранслируем n_o-угольник n_d раз снова по формуле (4). В результате получим $\omega_t \equiv r_1\cos\mu_1 - R_1 = 0$, где

$$\mu_1 = \frac{8}{n_o\pi}\sum_k (-1)^{k+1}\frac{\sin\left[(2k-1)\frac{n_o\theta_1}{2}\right]}{(2k-1)^2};$$

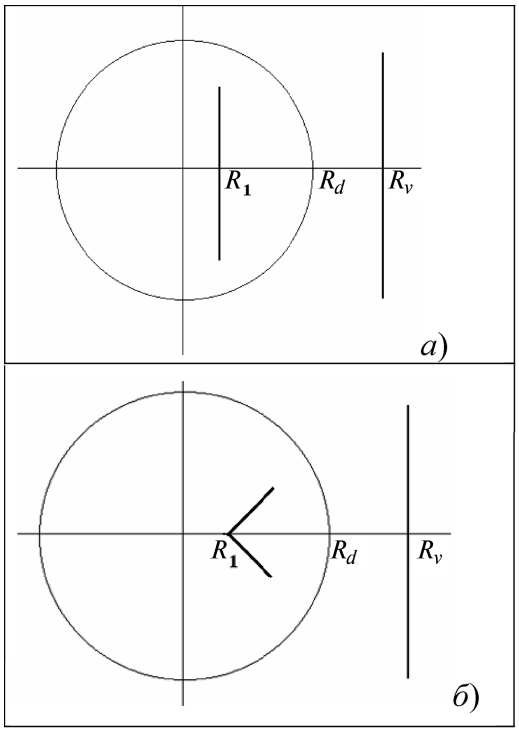


Рис. 2. Схема построения ГО транслируемого в правильном n_b-угольнике n_d раз:
а — для ГО в виде правильного n_o-угольника; *б* — для ГО в виде n_o-угольной звезды

$$\theta_1 = \frac{1 - \operatorname{sign}(x_s)}{2}\pi + \operatorname{arctg}\frac{y_s}{x_s} -$$

преобразования для правильного n_o-угольника, а

$$\mu_d = \frac{8}{n_d\pi}\sum_k (-1)^{k+1}\frac{\sin\left[(2k-1)\frac{n_d\theta}{2}\right]}{(2k-1)^2};$$

$$\begin{cases} x_s = r\cos\mu_d - R_d, \\ y_s = r\sin\mu_d \end{cases};$$

$r_1 = \sqrt{x_s^2 + y_s^2}$ — преобразования переноса начала координат в точку $x = R_d$ и трансляции n_d раз по окружности.

Для построения уравнения внешнего правильного n_b-угольника воспользуемся уравнением прямой $R_v - x \geqslant 0$ и формулой (4). В результате получим

$$\omega_v \equiv R_v - r\cos\mu_v = 0,$$

где $\mu_v = \frac{8}{n_b\pi}\sum_k (-1)^{k+1}\frac{\sin\left[(2k-1)\frac{n_b\theta}{2}\right]}{(2k-1)^2}$,

$$r = \sqrt{x^2 + y^2}.$$

Окончательно, нормализованное уравнение границы правильного n_o-угольника оттранслированного в правильном n_b-угольнике n_d раз имеет вид $\omega \equiv \omega_v \wedge_0 \omega_t = 0$ и является шестипараметрическим (n_o, n_b, n_d, R_1, R_d, R_v) семейством кривых. При этом следует отметить, что R-операция использовалась лишь один раз. Для сравнения рассмотрим простейший случай, когда $n_b = 3$, $n_o = 3$, $n_d = 3$. Применяя классический подход для построения функции ω, необходимо задать шесть опорных функций (шесть прямых) и применить семь R-операций.

Рассмотрим пример, когда правильные n_o-угольные звезды транслируются в правильном n_b-угольнике n_d раз (рис. 3, см. третью сторону обложки). При этом все построения выполним с помощью трех прямых: $f_1 = \dfrac{(-y + k(x - R_1))}{\sqrt{1+k^2}} \geqslant 0$, $f_2 = \dfrac{(y - k(-x + R_1))}{\sqrt{1+k^2}} \geqslant 0$, где $k = \dfrac{10\sin(\pi/n_o)}{10\cos(\pi/n_o) - 4}$ и $f_3 = R_v - x \geqslant 0$ (рис. 2, *б*).

Для построения уравнения n_o-угольной звезды проконъюнктируем нормализованные уравнения двух прямых $\omega_t \equiv f_1 \wedge_0 f_2 = 0$ и воспользуемся формулой (4). Затем, перенося начало координат в точку $x = R_d$, оттранслируем звезду n_d раз снова по формуле (4). В результате получим

$$\omega_t \equiv \left(\frac{(-r_1\sin\mu_1 + k(r_1\cos\mu_1 - R_1))}{\sqrt{1+k^2}}\right) \wedge_0$$

$$\wedge_0 \left(\frac{(r_1\sin\mu_1 - k(-(r_1\cos\mu_1 + R_1)))}{\sqrt{1+k^2}}\right) = 0,$$

где $\theta_1 = \dfrac{1 - \mathrm{sign}(x_s)}{2}\pi + \mathrm{arctg}\dfrac{y_s}{x_s}$, $\mu_1 = \dfrac{8}{n_o\pi}\sum\limits_k (-1)^{k+1} \times \dfrac{\sin\left[(2k-1)\dfrac{n_o\theta_1}{2}\right]}{(2k-1)^2}$ — преобразования для построения n_o-угольной звезды;

$$\mu_d = \frac{8}{n_d\pi}\sum_k (-1)^{k+1}\frac{\sin\left[(2k-1)\dfrac{n_d\theta}{2}\right]}{(2k-1)^2};$$

$$\begin{cases} x_s = r\cos\mu_d - R_d, \\ y_s = r\sin\mu_d \end{cases};$$

$r_1 = \sqrt{x_s^2 + y_s^2}$ — преобразования переноса начала координат в точку $x = R_d$ и трансляции звезды n_d раз по окружности.

Для построения уравнения внешнего правильного n_b-угольника воспользуемся, как и в предыдущем случае, уравнением прямой $R_v - x \geqslant 0$ и формулой (4). В результате получим $\omega_v \equiv R_v - r\cos\mu_v = 0$, где $\mu_v = \dfrac{8}{n_b\pi}\sum\limits_k (-1)^{k+1}\dfrac{\sin\left[(2k-1)\dfrac{n_b\theta}{2}\right]}{(2k-1)^2}$, $r = \sqrt{x^2 + y^2}$.

Окончательно, нормализованное уравнение границы n_o-угольной звезды, оттранслированной в правильном n_b-угольнике n_d раз, имеет вид $\omega \equiv \omega_v \wedge_0 \omega_t = 0$ и также является шестипараметрическим (n_o, n_b, n_d, R_1, R_d, R_v) семейством кривых. Отметим, что в данном случае R-операция использовалась дважды.

Если область Σ_0 несимметрична относительно оси абсцисс, но при циклическом тиражировании необходимо сохранить ее ориентацию относительно декартовой системы координат xOy, то функция $\omega(x, y)$, нормализованная на границе области $\Omega = [\omega(x, y) \geqslant 0]$, может быть представлена формулой

$$\omega(x, y) \equiv \left[\sigma_0(x', y') \wedge_\alpha \cos\left(\frac{n\theta}{2}\right)\right] \vee_\alpha$$

$$\vee_\alpha \left[\sigma_0(x'', y'') \wedge_\alpha \cos\left(\frac{n}{2}\left(\theta - \frac{2\pi}{n}\right)\right)\right] = 0, \qquad (6)$$

где $\begin{cases} x' = (r\cos\mu(n\theta) - r_0)\cos\alpha_k - r\sin\mu(n\theta)\sin\alpha_k, \\ y' = (r\cos\mu(n\theta) - r_0)\sin\alpha_k + r\sin\mu(n\theta)\cos\alpha_k \end{cases}$

$$\begin{cases} x'' = \left(r\cos\mu\left(n\left(\theta - \dfrac{2\pi}{n}\right)\right) - r_0\right)\cos\alpha_k - \\ - r\sin\mu\left(n\left(\theta - \dfrac{2\pi}{n}\right)\right)\sin\alpha_k, \\ y'' = \left(r\cos\mu\left(n\left(\theta - \dfrac{2\pi}{n}\right)\right) - r_0\sin\alpha_k + \right. \\ \left. + r\sin\mu\left(n\left(\theta - \dfrac{2\pi}{n}\right)\right)\cos\alpha_k \right. \\ \alpha_k = \theta - \mu(n\theta). \end{cases}$$

Преобразование координат для построения уравнений, соответствующих конечное число раз протранслированным вдоль прямой ГО. В основе данного подхода лежит преобразование координат с заранее определенным числом $n < \dfrac{l}{a} + 1$ ГО на отрезке $\mu^*(x, h) = (x \wedge_1 \mu(x, h)) \wedge_1^n ((x - l)(-1)^{n-1})$, где $\wedge_1^n = \dfrac{1}{2}(x + y + (-1)^{n-1}|x - y|)$. В результате получаем

$$\omega(x, y) = \sigma_0(\mu^*(x, h), y) = 0. \qquad (7)$$

На рис. 4 (см. третью сторону обложки) приведены примеры перфорации отверстий в виде кругов и прямоугольников, где функция $\omega(x, y)$ построена с использованием преобразования координат $\mu^*(x, h)$. При этом $\omega \equiv \left(\left(\frac{4-y^2}{4}\right) \wedge_0 \wedge_0 \left(\frac{(2+x)(25-x)}{27}\right)\right) \wedge_0 \overline{\omega_1} \geqslant 0$.

Для круговых отверстий $\omega_1 = \frac{1-(\mu^*(x,h))^2-y^2}{2} \geqslant 0$, $h = 3$, $l = 12$.

Для прямоугольных отверстий $\omega_1 = (0{,}25 - (\mu^*(x, h))^2) \wedge_0 (0{,}25 - y^2) \geqslant 0$, $h = 2$, $l = 12$.

Преобразование координат для построения уравнений, соответствующих конечное число раз протранслированным вдоль дуги ГО. Преобразование координат, осуществляющее трансляцию элемента на интервале φ с шагом θ_h, имеет вид

$$\mu^*(\theta_1, \theta_h) = \left(\left(\theta_1 - \frac{\theta_h}{2}\right) \wedge_1 \mu\right) \wedge_1 \wedge_1 \left(\varphi - \left(\theta_1 + \frac{\theta_h}{2}\right)\right), \quad (8)$$

где $\mu(\theta_1, \theta_h) = \frac{4\theta_h}{\pi^2} \sum_k (-1)^{k+1} \frac{\sin\left[(2k-1)\left(\theta_1 - \frac{\theta_h}{2}\right)\frac{\pi}{\theta_h}\right]}{(2k-1)^2}$,

$\theta_1 = \theta - \pi + \frac{\varphi}{2}$, $n_d = \frac{\varphi}{\theta_h}$ — число элементов на интервале.

На рис. 5 (см. четвертую сторону обложки) приведены примеры перфорации отверстий в виде треугольников и прямоугольников с функцией $\omega(x, y)$, построенной при использовании преобразования координат (8).

Отметим, что нормализованное уравнение $\omega(x, y) = 0$ границы правильного n_o-угольника, оттранслированного в правильном n_b-угольнике n_d раз на интервале трансляции φ ($n_d = \frac{\varphi}{\theta_h}$ — число элементов на интервале, θ_h — шаг трансляции) является семипараметрическим (n_o, n_b, φ, θ_h, R_1, R_d, R_v) семейством кривых. При этом R-операция $\wedge_0$ использовалась один раз, а $\wedge_1$ — дважды при построении преобразования $\mu^*(\theta_1, \theta_h)$.

Заметим, что все описанные выше методы справедливы и для ГО в nD ($n > 2$).

Рассмотрим конструктивную схему типичного водоводяного реактора, активная зона которого собирается из большого числа топливных кассет [7]. Кассеты представляют собой шестигранные кожухи, в которых размещены тепловыделяющие элементы (рис. 6, см. четвертую сторону обложки). Построим уравнение топливной кассеты с 91 тепловыделяющим элементом (ТВЭЛ) и раздвинутой треугольной упаковкой, которую иногда называют шахматной [7]. Для построения шестигранного кожуха воспользуемся методикой из примеров, проиллюстрированных на рис.1. Тогда $f_b = r_k - x_1 \geqslant 0$, где

$$x_1 = r\cos\mu_1;$$

$$\mu_1 = \frac{8}{n_0\pi} \sum_k (-1)^{k+1} \frac{\sin\left[(2k-1)\frac{n_0\theta}{2}\right]}{(2k-1)^2}.$$

Для построения треугольной упаковки ТВЭЛов зададим $f_1 = \frac{R^2 - \mu_x^2 - \mu_y^2}{2R} \geqslant 0$, где $\mu_x = \frac{4h_x}{\pi^2} \sum_k (-1)^{k+1} \frac{\sin\left[(2k-1)\frac{\pi x}{h_x}\right]}{(2k-1)^2}$, $\mu_y = \frac{4h_y}{\pi^2} \sum_k (-1)^{k+1} \times \frac{\sin\left[(2k-1)\frac{\pi x}{h_y}\right]}{(2k-1)^2}$, и $f_2 = \frac{R^2 - \mu_{x_1}^2 - \mu_{y_1}^2}{2R} \geqslant 0$,

где $\mu_{x1} = \frac{4h_x}{\pi^2} \sum_k (-1)^{k+1} \frac{\sin\left[\frac{(2k-1)\pi(x - h_x/2)}{h_x}\right]}{(2k-1)^2}$,

$$\mu_{y1} = \frac{4h_y}{\pi^2} \sum_k (-1)^{k+1} \frac{\sin\left[\frac{(2k-1)\pi(y - h_y/2)}{h_x}\right]}{(2k-1)^2}.$$

Тогда нормализованное уравнение топливной кассеты имеет вид $\omega \equiv f_b \wedge_0 (\overline{f_1 \vee_0 f_2}) = 0$.

На рис. 7 (см. четвертую сторону обложки) приведена картина линий уровня функции $\omega(x, y)$ при следующих значениях буквенных параметров:

$$R = 0{,}2;\ h_x = 2{,}32;\ h_y = 1{,}35;\ n_0 = 6;\ r_k = 6{,}7.$$

Заметим, что R-операция применена лишь дважды для трех опорных функций, в то время как при классическом подходе потребовалось бы 94 опорные функции и 93 R-операции.

В работе [8] рассматривается интересный тип симметрии в кристаллической структуре редкоземельных металлоорганических соединений, используемых при хранении водорода (рис. 7, см. четвертую сторону обложки). Построим уравнение этой структуры.

Семь больших сфер описываем следующим образом:

$$f_1 = \left(\frac{r_1^2 - x^2 - y^2}{2r_1}\right) \vee_0 \left(\frac{r_1^2 - x_1^2 - y_1^2}{2r_1}\right) \geqslant 0,$$

где $\begin{cases} x_1 = \rho\cos\mu_1 - r_{k1}, \\ y_1 = \rho\sin\mu_1 \end{cases}$

$\mu_1 = \frac{8}{n_0\pi}\sum_k(-1)^{k+1}\frac{\sin\left[(2k-1)\frac{n_0\theta}{2}\right]}{(2k-1)^2}$, а трансляцию триад маленьких сфер осуществим по формулам

$$f_2 = \left(\frac{r_2^2 - \left(x_2 - 2r_2\cos\frac{\pi}{6}\right)^2 - y_2^2}{2r_2}\right) \vee_0$$

$$\vee_0\left(\frac{r_2^2 - x_2^2 - (y_2 + r_2)^2}{2r_2}\right) \vee_0 \left(\frac{r_2^2 - x_2^2 - (y_2 - r_2)^2}{2r_2}\right) \geqslant 0,$$

где $\begin{cases} x_2 = \rho_1\cos\mu_2 - r_{k2}; \\ y_2 = \rho_1\sin\mu_2 \end{cases}$ $\quad \begin{cases} \rho_1 = \sqrt{x_1^2 + y_1^2} \\ \theta_1 = \operatorname{arctg}\frac{y_1}{x_1} \end{cases};$

$$\mu_2 = \frac{8}{n_0\pi}\sum_k(-1)^{k+1}\frac{\sin\left[\frac{(2k-1)n_0(\theta_1 - \pi/2)}{2}\right]}{(2k-1)^2}.$$

Окончательно, нормализованное уравнение поверхности рассмотренной кристаллической структуры имеет вид $\omega \equiv f_1 \vee_0 f_2 = 0$. Вычисления проводились при следующих значениях буквенных параметров:

$r_1 = 0{,}8;\ r_2 = 0{,}16;\ r_1 = 0{,}8;\ r_{k1} = 2{,}2;\ r_{k2} = 1{,}15.$

Выводы и перспективы

В отличие от классической схемы применения RFM предлагаемый метод позволяет включать в вычислительные алгоритмы геометрическую информацию о ГО, сочетающих в себе различные типы симметрий. При этом построение нормализованных уравнений ГО не требует многократного использования R-операций, а осуществляется путем введения специальных преобразований координат, что позволяет существенно сократить вычислительные затраты во всех приложениях, требующих преобразования геометрической информации в аналитическую. Следует отметить новые преобразования координат, позволяющие осуществлять трансляцию ГО на конечных интервалах. Такие подходы особенно важны при решении проблемы автоматизации в твердотельном моделировании (*solid modelling*), геометрическом проектировании (*geometrical design*) и при решении краевых задач математической физики. Особенно много научно-технических решений, опирающихся на исследования полей различной физической природы, связано с повышением прочности, надежности, устойчивости, долговечности, добротности изделий и конструкций при наличии симметрии. Здесь можно назвать перфорированные пластины и оболочки (фюзеляжи самолетов, корпуса судов и космических кораблей, ослабленные иллюминаторами); пилы, фрезы, зубчатые передачи; решетки в технике сверхвысоких частот и гидродинамических конструкциях; роторы турбин и бланкеты тепловыделяющих элементов ядерных реакторов.

Новые методы построения нормализованных уравнений симметричных ГО и ГО, транслируемых на конечных интервалах, легко адаптируются к любому программному обеспечению и удобны при использовании метода стандартных примитивов [6]. Их численная реализация в системах ПОЛЕ и РАНОК [1, 5] подтвердила многократное сокращение временных затрат.

Список литературы

1. **Рвачев В. Л.** Теория R-функций и некоторые ее приложения. Киев: Наук. думка, 1982. 566 с.

2. **Рвачев В. Л., Шапиро В., Шейко Т. И.** Метод R-функций (RFM) в краевых задачах с геометрической и физической симметрией // Математичні методи та фізико-механічні поля. 1998. 41. № 1. С. 146—159.

3. **Рвачев В. Л., Шапиро В., Шейко Т. И.** Применение метода R-функций к построению уравнений локусов, обладающих симметрией // Электромагнитные волны и электронные системы. 1999. № 4. Т. 4. С. 4—20.

4. **Семерич Ю. С., Толок А. В., Шейко Т. И.** Построение симметричных функций для симметричных чертежей // Вестник Запорож. гос. ун-та. 2001. № 2. С. 83—98.

5. **Максименко-Шейко К. В., Мацевитый А. М., Толок А. В., Шейко Т. И.** R-функции и обратная задача аналитической геометрии в трехмерном пространстве / // Информационные технологии. 2007. № 10. С. 23—32.

6. **Максименко-Шейко К. В., Мацевитый А. М., Шейко Т. И.** Автоматизация построения уравнений геометрических объектов в методе R-функций // Кибернетика и системный анализ. 2006. № 2. С. 148—157.

7. **Петухов Б. С., Генин Л. Г., Ковалев С. А.** Теплообмен в ядерных энергетических установках. М.: Атомиздат, 1974. 367 с.

8. **Fang Q. R., Zhu G. S., Jin Z., Ji Y. Y., Xue M., Yang H., Qiu S. L**. Mesoporous Metal-Organic Framework with Rare etb Topology for Hydrogen Storage and Dye Assembly // Angew. Chem. Int. Ed. 2007. **46.** P. 1—6.

УДК 004.421, 004.932.2, 004.272.25, 004.92

А. П. Кудряшов, мл. науч. сотр.,
Институт автоматики и процессов управления РАН, г. Владивосток,
e-mail: kudryshovA@iacp.dvo.ru

Реконструкция трехмерных сцен городской обстановки[1]

Представлен метод реконструкции трехмерных сцен городской обстановки по некалиброванной последовательности фотоизображений. Предлагаемый подход основан на моделировании полигональных объектов по пространственным отрезкам, полученным в результате предварительной векторизации исходных изображений, их калибровки и работы алгоритма сопоставления линий на изображениях. Метод существенно использует ортогональность и параллельность ребер и граней объектов сцены.

Ключевые слова: *трехмерная реконструкция, сцены городской обстановки, эпилиния, эпиполярная геометрия, векторизация, сопоставление.*

Введение

Настоящая работа является развитием ранее опубликованных автором работ [1, 2], где решалась задача калибровки камер с использованием найденных точек схода (*vanishing points*) и точечных соответствий на изображениях сцен городской обстановки [3]. Там же решалась задача сопоставления прямолинейных отрезков-ребер зданий для калиброванных изображений с применением трифокальной геометрии и методики корреляционного сравнения пиксельных образов в целях последующей пространственной реконструкции сцены.

В последние годы исследователями в области компьютерного зрения много внимания было уделено решению задачи реконструкции пространственных сцен по некалиброванным изображениям. Основными методами реконструкции являются:

- представление сцены в виде текстурированной триангуляционной сетки, построенной по восстановленным трехмерным точкам [8, 9];
- представление восстановленных точечных особенностей в виде вокселей в трехмерном пространстве сцены [10];
- построение полигональной модели по сгруппированным восстановленным трехмерным линиям [5—7];
- интерактивное моделирование с полуавтоматическим подбором геометрических примитивов (параллелепипедов, цилиндров, пирамид, плоскостей и т. д.) по восстановленным особенностям [11—13].

В целом предлагаемая в настоящей статье методика, отталкиваясь от известных идей, в том числе изложенных в работах [5—7], и базируясь на оригинальных алгоритмах, направлена на автоматическое построение модели сцены с минимизацией влияния ошибок, неизбежно присущих векторизации исходных изображений и определению особенностей. В отличие от упомянутых аналогов в данном методе наряду с параллельностью/ортогональностью существенно используется связность отрезков, что позволяет избежать ошибок при вычислении ориентации плоскостей объектов, упростить реконструкцию и получить более точную трехмерную модель рассматриваемой сцены.

Для оценки эффективности предложенной методики проводились вычислительные эксперименты как с модельными, так и с реальными сценами.

Построение отрезков в трехмерном пространстве

Восстановление направления сопоставленных отрезков. В результате работы алгоритма сопоставления отрезков [1, 2] мы получаем множество сопоставленных отрезков. Но при этом нельзя утверждать, что начальная точка сопоставленного отрезка на одном изображении соответствует начальной точке соответствующего отрезка на другом изображении. Рассмотрим подобную ситуацию для двух изображений (рис. 1), затем расширим схему восстановления для произвольного числа изображений.

Из рис. 1 видно, что хотя отрезок (p_{21}^l, p_{22}^l) соответствует отрезку (p_{21}^r, p_{22}^r), точка p_{11}^l не соответствует точке p_{11}^r и т. д. Восстанавливать соответствие точек будем, используя эпиполярные ограничения [4]. Построим эпилинию l для точки p_{21}^l по формуле

$$l = F^{\mathrm{T}} p_{21}^l, \qquad (1)$$

где F — фундаментальная матрица между левым и правым изображениями.

Полученная эпилиния пересечет линию, содержащую отрезок (p_{21}^r, p_{22}^r) в точке P. Если расстояние от точки P до точки p_{21}^r больше, чем расстояние от точки P до точки p_{22}^r, меняем точки p_{21}^r и p_{22}^r местами, т. е. меняем направление данного отрезка. Если число изображений больше двух, то в качестве исходных направлений отрезков берутся

[1] Работа выполнена при финансовой поддержке РФФИ (проект 08-07-00113-а).

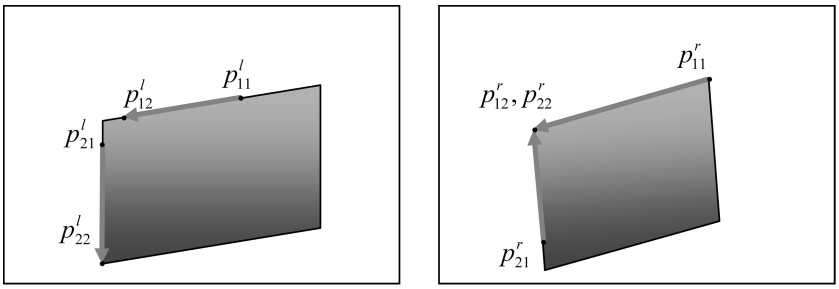


Рис. 1. Два изображения с двумя парами сопоставленных отрезков

направления на первом изображении, а направления на последующих изображениях проверяются и в случае необходимости меняются. На длинной последовательности могут существовать отрезки, отсутствующие на первом изображении, но присутствующие на изображениях, начиная со второго, третьего и т. д. Поэтому будем рассматривать отрезки на втором изображении, которых нет на первом, и обрабатывать их аналогичным способом.

Сопоставление угловых точек на сопоставленных отрезках. На рис. 1 видно, что точка p^l_{22}, p^r_{12} — является угловой, т. е. точкой соприкосновения двух отрезков. Но на левом изображении, по какой-либо причине (например, ошибка векторизации, перекрытие посторонним объектом и т. п.) эта угловая точка не образовалась. Однако известно, что ее прообразами являются точки p^l_{22} и p^l_{11} (рис. 2). Поэтому мы можем восстановить эту угловую точку на левом изображении путем нахождения пересечения линий, содержащих отрезки (p^l_{11}, p^l_{12}) и (p^l_{21}, p^l_{22}). Для этого необходимо решить систему линейных уравнений

$$\begin{cases} \dfrac{x_{p^l_{11}} - x}{x_{p^l_{11}} - x_{p^l_{12}}} = \dfrac{y_{p^l_{11}} - y}{y_{p^l_{11}} - y_{p^l_{12}}}; \\ \dfrac{x_{p^l_{21}} - x}{x_{p^l_{21}} - x_{p^l_{22}}} = \dfrac{y_{p^l_{21}} - y}{y_{p^l_{21}} - y_{p^l_{22}}}, \end{cases} \qquad (2)$$

где точка P с координатами (x, y) — точка, которая соответствует угловой точке p^r_{12} (и точке p^r_{22}) (см. рис. 1).

Для последовательности изображений на каждом изображении осуществляется поиск угловых точек, и в случае необходимости отрезки на других изображениях достраиваются пересечением.

Сопоставление свободных точек на сопоставленных отрезках. Свободными точками назовем такие точки, которые принадлежат только одному отрезку, т. е. не являются угловыми. Очевидно, что точки p^l_{11} и p^l_{21} являются прообразами точек p^r_{11} и p^r_{21}, но им не соответствуют (рис. 3). Достроить отрезки, содержащие свободные точки, до максимально видимой длины можно, опять же используя эпиполярные ограничения. Построим эпилинию l для точки p^l_{21} по формуле (1) и найдем пересечение полученной эпилинии с линией, содержащей отрезок (p^r_{21}, p^r_{22}), обозначим точку пересечения p (рис. 3). Если отрезок (p, p^r_{22}) длиннее отрезка (p^r_{21}, p^r_{22}), то приравниваем координаты p^r_{21} к координатам найденной точки p. Иначе, строим эпилинию на левом изображении и находим пересечение эпилинии с линией, содержащей соответствующий отрезок. В случае, когда эпилиния параллельна отрезку, с которым находится пересечение, точка пересечения уходит в бесконечность. Поэтому проверяем угол между эпилинией и отрезком, и если он меньше определенного значения, то пересечение искать не будем.

В результате получим множество пар соответствующих отрезков (рис. 4), у которых все соответствующие точки сопоставлены друг другу.

Нахождение трехмерных координат концевых точек отрезков. При наличии откалиброванной пары изображений и сопоставленных точек проекций p^l и p^r на эти изображения задача восстановления точки P в трехмерном пространстве сводится к нахождению пересечения двух лучей: $R^l = \overrightarrow{O^l p^l}$ и $R^r = \overrightarrow{O^r p^r}$ (рис. 5). Но на практике эти лучи могут не пересекаться из-за погрешностей калибровки, неточных соответствий точек и т. п.

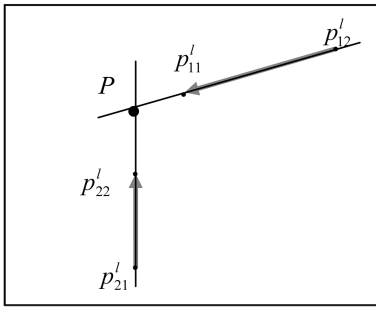


Рис. 2. Восстановление соответствующей угловой точки

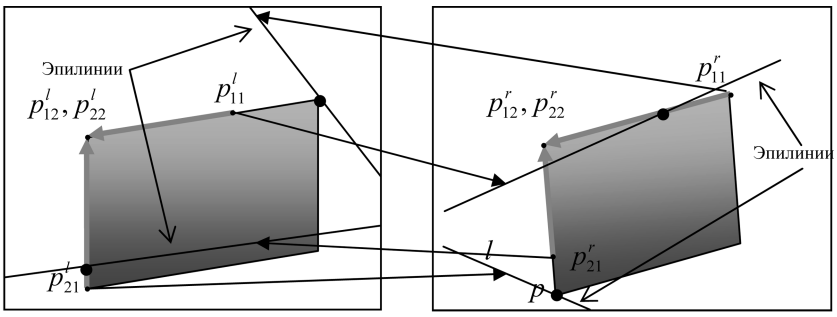


Рис. 3. Достройка отрезков со свободными точками

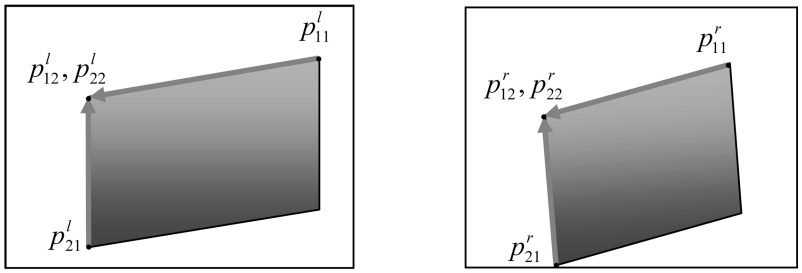


Рис. 4. Два изображения с двумя парами сопоставленных отрезков с сопоставленными точками

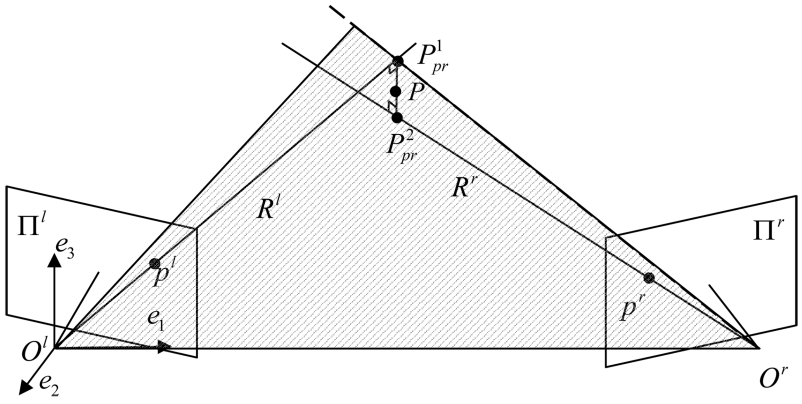


Рис. 5. Проецирование луча R^r на плоскость $O^lp^lO^r$

Точку P находим следующим образом: спроецируем луч R^r на плоскость $O^lp^lO^r$ и получим:

$$R^r_{pr} = M \times \begin{pmatrix} x_{R^r} \\ y_{R^r} \\ z_{R^r} \\ 1 \end{pmatrix}, \text{ где } M = \begin{pmatrix} x_{e_1} & x_{e_2} & x_{e_3} & x_{O^l} \\ y_{e_1} & y_{e_2} & y_{e_3} & y_{O^l} \\ z_{e_1} & z_{e_2} & z_{e_3} & z_{O^l} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (3)$$

где $e_i(x_{e_i}, y_{e_i}, z_{e_i})$, $i = 1, 2, 3$, — базисы новой системы координат с центром в O^l, где

$$\begin{cases} e_1 = \overrightarrow{O^lO^r}; \\ e_3 = e_1 \times \overrightarrow{O^lp^l}; \\ e_2 = e_1 \times e_3. \end{cases} \quad (4)$$

Точку $P^l_{pr}(x, y, 0)$ находим, решая систему линейных уравнений

$$\begin{cases} x = 0; \\ \dfrac{x_{p^r_{pr}}}{x_{p^r_{pr}} - x_{O^r_{pr}}} = \dfrac{y_{p^r_{pr}} - y}{y_{p^r_{pr}} - y_{O^r_{pr}}}, \end{cases} \quad (5)$$

где $p^r_{pr} = M \times \begin{pmatrix} x_{p^r} \\ y_{p^r} \\ z_{p^r} \\ 1 \end{pmatrix}$; $O^r_{pr} = M \times \begin{pmatrix} x_{O^r} \\ y_{O^r} \\ z_{O^r} \\ 1 \end{pmatrix}$.

Тогда координаты точки P^1 в системе координат камеры вычисляем по формуле

$$P^1 = M^{-1} \times (0\ y\ 0\ 1). \quad (6)$$

Аналогичным образом найдем точку P^2, а точку P найдем, усредняя координаты точек P^1 и P^2. Таким образом, определив трехмерные координаты концевых точек каждого отрезка, получаем трехмерное положение этих отрезков в сцене.

Нахождение дополнительных отрезков по первичной векторизации. Число сопоставленных отрезков всегда меньше, чем отрезков, построенных в ходе векторизации изображений [2, табл. 1]. Причинами являются неточная векторизация, перекрытие отрезков другими объектами и т. д. Еще одна причина — какой-то отрезок присутствует только на одном или двух видах, а на остальных он не наблюдаем. Часть из этих линий можно найти и использовать на этапе трехмерной реконструкции (рис. 6).

Попарно переберем все точки сопоставленных отрезков и проверим, существует ли хоть в одном наборе исходных отрезков каждого изображения такой отрезок, который бы соединял эти две точки. Понятно, что при этом необходимо исключить из рассмотрения уже существующие сопоставленные отрезки. Если такой отрезок, соединяющий пару точек, найден, то добавляем его к общему списку сопоставленных

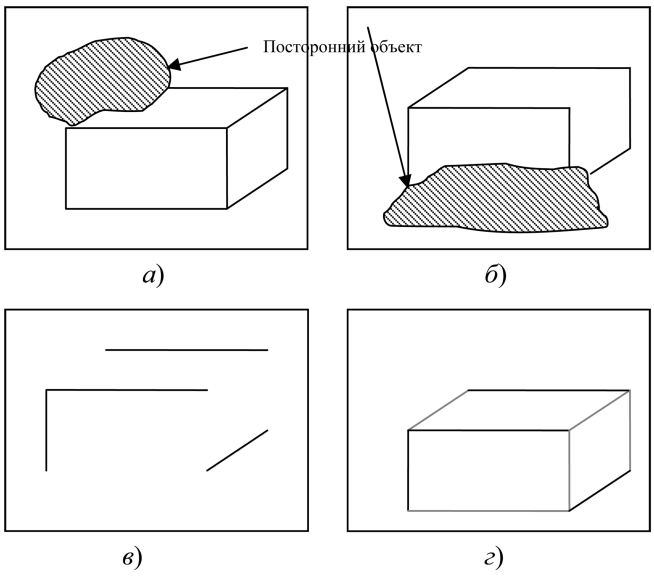


Рис. 6. Трехмерная реконструкция:
а и *б* — исходная векторизация на паре снимков; *в* — результат работы алгоритма сопоставления отрезков; *г* — результат работы алгоритма достройки отрезков

Результаты работы алгоритма нахождения дополнительных отрезков

Сцена	Общее число отрезков в сцене	Число сопоставленных отрезков	Число достроенных отрезков
Модельная "Город"	836	376	134
Реальная "ИАПУ"	45	27	11
Реальная "Дом"	24	23	6
Реальная "Фуникулер"	108	48	26

отрезков на всех видах. В таблице представлены результаты работы данного алгоритма на некоторых модельных и реальных сценах.

Построение полигональных объектов

Группировка и фильтрация отрезков. Среди множества отрезков, трехмерное положение которых было вычислено на предыдущих этапах, существует определенное число ложных и неточно восстановленных отрезков. Их число зависит от точности калибровки, векторизации и работы алгоритма сопоставления отрезков. Поэтому для удаления ложных векторов воспользуемся тем фактом, что мы восстанавливаем сцену городской обстановки, где основными объектами являются дома с вертикальными стенами, плоскими крышами и большинство углов, присутствующих в сцене, являются прямыми. Отсортируем все отрезки на две группы: вертикальные, горизонтальные. Вертикальными отрезками будем считать отрезки, угол между которыми и осью *OZ* не превышает заданного значения. Аналогичным образом отберем отрезки, горизонтальные по отношению к плоскости *XY*. Остальные отрезки исключаем из дальнейшего рассмотрения.

Построение полигонов, образующих стены зданий. Полигональное восстановление сцены начинается с полигонов, образующих стены. Полигон стены состоит из четырех точек и четырех соединяющих их отрезков — двух вертикальных и двух горизонтальных (рис. 7).

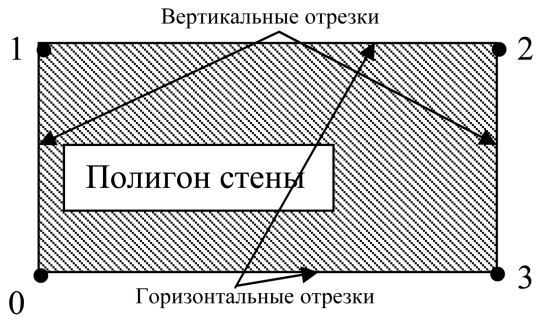


Рис. 7. Общее представление полигона, который образует стену здания; цифрами обозначены угловые точки, образующие полигон

Поскольку полученные на предыдущих этапах 3D-отрезки не претендуют на полное описание трехмерной сцены, очевидно, что для большинства искомых полигонов стен будут отсутствовать некоторые элементы. Поэтому предлагается нижеследующая рекуррентная схема реконструкции, которая обеспечивает восстановление всех возможных полигонов, образующих стены, а также построение дополнительных отрезков, необходимых для построения полигонов крыш. Учитывая, что полигон крыши образуют четыре отрезка, проанализируем все возможные ситуации.

Существуют все отрезки или отсутствует отрезок 0—3. Это самый простой случай. Будем перебирать горизонтальные отрезки, поскольку вертикальный отрезок образует два полигона стен, а горизонтальный только один, поэтому можно найти полигоны за один проход. Для каждого горизонтального отрезка находим два вертикальных отрезка, имеющих с ним общие точки, обозначенные на рис. 7 номерами 1 и 2. Затем проверяется, существует ли среди горизонтальных отрезков отрезок 0—3. Если такого отрезка не существует — добавляем. Все рассмотренные горизонтальные отрезки исключаем из дальнейшего поиска.

Нет одного вертикального отрезка. Для всех горизонтальных отрезков и найденного вертикального отрезка с общей точкой, обозначенной номером 1 или 2, найдем все горизонтальные линии, имеющие общую с ней общую точку 0 или 3. В идеальной сцене отрезки 0—3 и 1—2 должны быть параллельны, но на практике это не так. Спроецируем отрезок 1—2 и найденный отрезок на плоскость *XY* и определим угол между ними. Если он близок к нулю, то считаем этот отрезок искомым, а его концевые точки точками 0—3. Несуществующий отрезок 0—1 или 3—2 добавляем в список вертикальных отрезков. Все рассмотренные горизонтальные отрезки исключаем из дальнейшего поиска.

Нет одного вертикального и одного горизонтального отрезка. Находим горизонтальный и вертикальный отрезки, имеющие одну общую точку. Таким образом, получаем координаты трех точек, а четвертую точку вычисляем, используя гипотезу о прямоугольности полигона. А для того, чтобы найти текстурные координаты этой точки, т. е. координаты точки на изображениях, необходимо спроецировать эту точку на плоскость изображения каждой камеры. Поскольку калибровка не идеальна, то точка, спроецированная на плоскость изображения, может неточно соответствовать реальной, поэтому среди данных исходной векторизации находим наиболее близкую точку к спроецированной и считаем ее искомой.

Таким образом, по восстановленным в трехмерном пространстве отрезкам можно восстано-

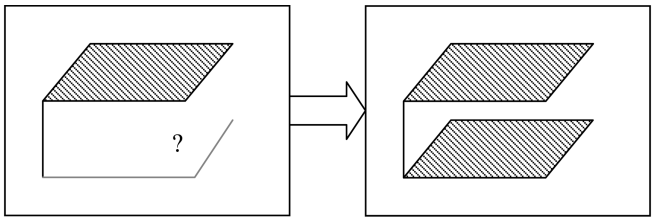


Рис. 8. Достройка неполного контура основания здания до полного с использованием достроенного контура для крыши здания. Полигоны стен не показаны

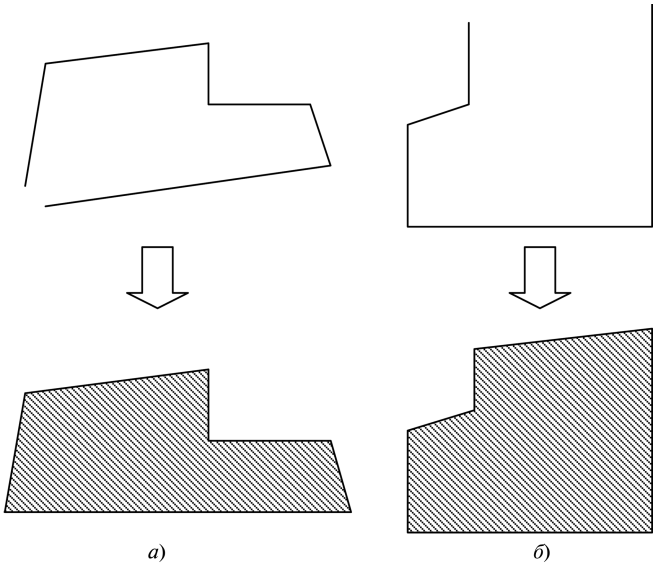


Рис. 9. Варианты замыкания незамкнутых контуров

вить полигоны, образующие стены зданий, даже если нет части отрезков, образующих контур здания. Описанная процедура выполняется рекурентно, пока не будут восстановлены все возможные полигоны.

Построение полигонов, образующих крыши или основания зданий. После восстановления полигонов стен выполняется процедура нахождения полигонов, образующих крыши и основания. Она состоит в анализе всех горизонтальных отрезков, в том числе и тех, которые были построены на этапе построения полигонов стен. Все соединенные друг с другом горизонтальные отрезки объединяются в контур, который в дальнейшем оптимизируется. Рассмотрим наиболее типичные варианты.

Замкнутый контур. Самый простой вариант — сразу записываем как полигон, образующий крышу или основание.

Существуют только два соединенных отрезка. В этом случае надо проверить, не существует ли еще один более полный контур для этого здания, поскольку для каждого здания существуют два горизонтальных контура — крыша и основание. Если такой контур найден, то достраиваем рассматриваемый контур до полного (рис. 8).

Если построить указанным образом контур не удалось, есть два варианта: соединить крайние точки отрезков и получить треугольный контур или не соединять их. В первом случае есть вероятность получить ошибочный полигон, например, в случае Г-образного здания, а во втором случае здание не будет построено вообще. Поэтому решение — строить такой полигон или нет — принимает пользователь.

В контуре есть разрыв. В этом случае используем гипотезу об ортогональности углов здания. Если угол между несоединенными отрезками близок к прямому, то получаем соединение путем усреднения координат несоединенных точек (рис. 9, *а*). Если угол между несоединенными отрезками близок к 0° или 180°, то строим отрезок, их соединяющий (рис. 9, *б*).

Оптимизация полигонов. Поскольку из-за возникающих ошибок на этапах реконструкции отрезки в трехмерном пространстве могли восстановиться неточно, необходимо провести оптимизацию полученных полигонов. Используем гипотезу о том, что все полигоны стен вертикальны и прямоугольны, полигоны крыш и оснований имеют углы между отрезками, составляющими эти полигоны, равные 90°.

Оптимизацию начнем с полигонов, образующих крыши и основания. Усредним Z-координату для всех точек полигона. Найдем пару таких отрезков, угол между которыми наиболее близок к 90°. Это будет начальной парой отрезков, с которой начнем выпрямление углов. Начиная с этого угла, достроим отрезки так, чтобы все углы контура были прямыми. Всем точкам полигона основания присвоим координаты XY соответствующих точек крыши. Далее среди всех полигонов стен найдем отрезок, имеющий наибольшую длину — это будет высота здания. На это расстояние относительно крыши перенесем полигон основания. Таким образом, будет построено все здание, у которого все углы будут 90°, все стены будут вертикальны и прямоугольны, а крыши и основания зданий будут параллельны друг другу и земле (рис. 10).

Объединение полученных полигонов в объекты (здания). Для сохранения полученной сцены и возможной дальнейшей обработки отдельных

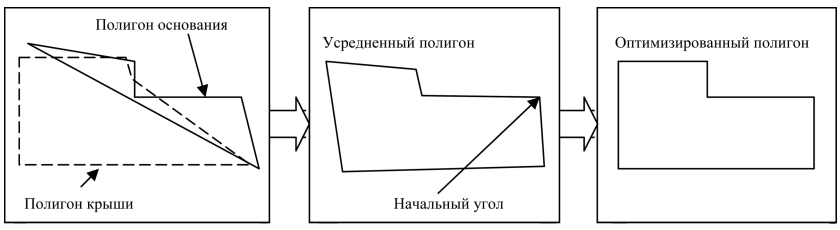


Рис. 10. Оптимизация контуров крыши и основания

объектов сцены удобно хранить данные об этих объектах сцены отдельно. Простейшим объектом назовем такой объект, который состоит только из основания, крыши и полигонов стен. Составной объект состоит соответственно из нескольких простых: основного нижнего сегмента и любого числа дополнительных сегментов. Примером может служить здание с несколькими уровнями, надстройками над крышами, треугольными или фигурными крышами.

Результаты

В ходе исследования были проведены вычислительные эксперименты как с модельными, так и с реальными сценами городской обстановки. Эксперименты показали, что при точной калибровке изображений (мерой точности служит отклонение эпилиний для контрольных точек, более подробно см. в [1, 2]) метод позволяет полностью восстановить трехмерную модель видимой области рассматриваемой сцены и нанести текстуру, взятую из исходных фотоизображений. Погрешности в геометрии, возникающие в результате неточной векторизации, эффективно устраняются оптимизацией полигонов с использованием гипотезы о прямоугольности и вертикальности зданий. Очевидно, что если какая-либо часть сцены не видна на двух и более изображениях сцены, восстановить эту область невозможно. Однако, если на изображениях присутствует крыша реконструируемого здания, ненаблюдаемые стены можно восстановить без нанесения на них текстуры. Примеры работы метода представлены на рис. 11, 12 (см. вторую сторону обложки).

Заключение

В статье представлен метод реконструкции трехмерных сцен городской обстановки по неограниченной некалиброванной последовательности фотоизображений, основанный на моделировании полигональных объектов по пространственным отрезкам. Проведенные вычислительные эксперименты позволяют говорить об эффективности и достаточной устойчивости работы метода. В дальнейшем планируется: оптимизация работы программы для ускорения обработки данных, повышение алгоритмической устойчивости к ошибкам начальной векторизации и калибровки изображений, алгоритмическое устранение помех при текстурировании объектов, а также расширение типов реконструируемых объектов.

Список литературы

1. **Бобков В. А., Роньшин Ю. И., Кудряшов А. П.** Сопоставление линий по трем видам пространственной сцены // Информационные технологии и вычислительные системы. 2006. № 2. С. 71—78.

2. **Бобков В. А., Роньшин Ю. И., Кудряшов А. П.** Идентификация линий по некалиброванным видам городской обстановки // Информационные технологии и вычислительные системы. 2007. № 1. С. 63—71.

3. **Кудряшов А. П.** Извлечение и сопоставление точечных особенностей. Электронный журнал "Исследовано в России". 2007. С. 1095—1104. http://zhurnal.ape.relarn.ru/articles/2007/104.pdf

4. **Luong Q. T. and Faugeras O. D.** The fundamental matrix: Theory, algorithms, and stability analysis // International Journal of Computer Vision. 1996. 17 (1). P. 43—76.

5. **Taillandier F., Deriche R**. Reconstruction of 3D linear primitives from multiple views for urban areas modelisation // ISPRS Commission III, Symposium. 2002. P. B-267.

6. **Leibowitz D., Criminisi A. and Zisserman A.** Creating architectural models from images // Proceedings EUROGRAPHICS. 1999. P. 39—50.

7. **Bailard C. and Zisserman A.** A plane-sweep strategy for the reconstruction of buildings from multiple images // ISPRS Congress and Exhibition. 2000. P. 105—112.

8. **Pollefeys M., Koch R., Vergauwen M., Deknuydt A. A., Gool L. J. V.** Three-dimentional Scene Reconstruction from images // Proc. SPIE-2000. Vol. 3958. P. 215—226.

9. **Zhang Z., Wu J., Zhang Y., Zhang J.** Multi-view 3d city model generation with image sequences. Vision techniques for digital architectural and archaeological archives. July 1—3 2003, Ancona, Italy, The International Archive of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2003. Vol. XXXIV, Part 5/W12. P. 262—267.

10. **Tomokazu Sato, Masayuki Kanbara, Naokazu Yokoya and Haruo Takemura.** 3-D Modeling of an Outdoor Scene by Multi-baseline Stereo Using a Long Sequence of Images, ICPR-2002, Part III. 2002. P. 581—584.

11. **Debevec P. E., Taylor C. J., Malik J.** Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach // SIGGRAPH'96. August 1996. P. 11—20.

12. **Rottensteiner F. and Schulze M.** Performance evaluation of a system for semi-automatic building extraction using adaptable primitives // Proceedings of the ISPRS Workshop "Photogrammetric Image analysis". 2003. Vol. 34; PART 3/W8. P. 47—52.

13. **Vezhnevets V., Konushin A., Ignatenko A.** Interactive image-based urban modelling // Photogrammetric Image Analysis. 2007. P. 63—69.

ДИСКУССИОННЫЙ КЛУБ

УДК 002.63

И. Д. Котляров, канд. экон. наук,
Северо-Западный институт печати
Санкт-Петербургского государственного
университета технологии и дизайна,
г. Санкт-Петербург, e-mail: lrpg@mail.ru

Сетевая публикация результатов диссертационных исследований

Предлагается усилить контроль за качеством результатов диссертационных исследований с помощью специализированного электронного архива публикаций аспирантов, докторантов и соискателей ученой степени. Сформулированы требования, которым должен удовлетворять такой архив, и предложен алгоритм его функционирования и механизм финансирования.

Ключевые слова: *результаты диссертационных исследований, электронный архив, апробация.*

Введение

Одной из важнейших задач при подготовке специалистов высшей квалификации (кандидатов и докторов наук) является обеспечение объективной и адекватной оценки достигнутых ими результатов научным сообществом. Для достижения этой цели Высшая аттестационная комиссия (ВАК РФ) регулярно формирует список ведущих российских рецензируемых журналов [2], в которых должны быть опубликованы результаты диссертационных исследований на соискание ученой степени кандидата и доктора наук (так называемый "список ВАК"). Соискатель степени кандидата наук (СК) обязан опубликовать в профильном (т. е. рекомендованным экспертным советом ВАК по соответствующей специальности) журнале не менее одной статьи; от соискателя докторской степени (СД) требуется не менее семи статей. Внесенные недавно изменения в требования к формированию "списка ВАК" [1] позволяют, в случае их неукоснительного соблюдения, устранить ряд проблем, связанных с публикацией научных результатов диссертантов (новый "список ВАК", сформированный в соответствии с этими требованиями, должен вступить в силу предположительно в конце 2009 г.):

1. Новые правила требуют обязательного рецензирования предоставляемых к публикации статей, высылки рецензий авторам, а также возможность запроса этих рецензий экспертными советами ВАК. В настоящее время весьма распространена ситуация, когда автор вместе со статьей должен также предоставить отзыв самостоятельно найденного им рецензента (сам журнал рецензированием не занимается). Изредка, но все еще встречается прием статей к публикации вообще без рецензирования. Наконец, "ваковские" журналы, издаваемые вузами, часто предлагают аспирантам, обучающимся в этих вузах, опубликовать в них статью без независимого рецензирования — только по отзыву своего научного руководителя. Для этих целей в данных журналах предусмотрены специальные разделы (например, "Страницы аспирантов и докторантов" в "Вестнике ИНЖЭКОНа"). Данное нововведение поставит барьер на пути этой практики.

2. Журналы из "списка ВАК" отныне обязаны иметь полнотекстовую сетевую версию в Интернете (при этом аннотации статей, сведения об авторах, ключевые слова и библиография должны быть в свободном доступе). Это существенно упрощает доступ к основным сведениям о научных результатах диссертантов (печатные версии журналов из "списка ВАК" издаются, за немногочисленными исключениями, малым тиражом, и поступают далеко не во все вузы и НИИ — даже профильные), и обеспечивает их более адекватную апробацию. Сейчас, к сожалению, не все "ваковские" журналы имеют полнотекстовые сетевые версии — в частности, она отсутствует у издаваемого весьма престижным Санкт-Петербургским государственным инженерно-экономическим университетом "Вестника ИНЖЭКОНа".

3. От журналов из "списка ВАК" теперь требуется предоставлять сведения об опубликованных в них статьях в систему Российского индекса научного цитирования, что позволяет оценивать востребованность результатов диссертационных исследований научным сообществом.

4. По косвенной информации можно сделать вывод о том, что в новом "списке ВАК" печатные и электронные издания будут уравнены в правах (до настоящего времени ВАК признавал статьи в зарегистрированных в Информрегистре электронных журналах в качестве обычных публикаций, а не публикаций в журнале из "списка ВАК", что в наше время представляется анахронизмом). Эта мера также будет способствовать упрощению доступа к результатам диссертационных исследований.

5. В соответствии с новыми требованиями журналам из "списка ВАК" запрещается взимать плату с аспирантов за публикацию их статей (ранее ВАК не рекомендовал взимание платы). Это требование очень важно: взимание платы за публикацию в ряде случаев ведет к профанации понятия "рецензируемый журнал", так как научное издание превращается из распространителя качественной и достоверной информации в продавца журнальных площадей и за плату публикует какие угодно материалы. Пример: громкий скандал с "Журналом научных публикаций аспирантов и докторантов" [5], который пропустил статью, представляющую собой случайно сгенерированный компьютерной программой текст на английском, переведенный на русский автоматическим переводчиком (рецензент при этом дал высокую оценку работе). Нужно отдать должное оперативности ВАК — в кратчайшие сроки было принято решение об исключении этого издания из списка рецензируемых журналов [4]. Другим примером могут быть "Известия РГПУ им. Герцена. Аспирантские тетради" (http://www.bookhouse.ru/?pagename = asptetr_price), которые предлагают услугу по срочной публикации статей (стоимость публикации одной авторской страницы составляет от 480 до 860 руб., в зависимости от того, насколько срочно автору нужно опубликовать свою работу). Разумеется, журналы, скорее всего, все равно продолжат взимать плату за публикацию (в частности, не запрещается ее взимание с соискателей и с докторантов; плату также можно брать за подписку, за авторский экземпляр и т. д.), поскольку даже уважаемым и престижным журналам эта плата нужна для окупаемости издания, однако новое требование ВАК, хочется надеяться, приведет к ликвидации откровенно коммерческих некачественных журналов.

На наш взгляд, необходимо разграничить требования, предъявляемые к публикации результатов собственно научных исследований и результатов, защищаемых в диссертациях (прежде всего — кандидатских). Научные журналы из "списка ВАК" в своей основе предназначены для публикации результатов научных исследований. Диссертация же, особенно кандидатская, является не только научной, но и квалификационной работой. Кроме того, не секрет, что статьи многих аспирантов не дотягивают до уровня, требуемого этими журналами (известно высказывание руководителя одного из ведущих российских журналов по экономике, который сказал, что не хочет добиваться включения своего журнала в "список ВАК", чтобы не засорять его аспирантскими статьями). При этом для повышения уровня аспирантских публикаций есть объективные препятствия: аспиранты — это, как правило, молодые люди, только что закончившие вуз и имеющие мало опыта как в проведении научных исследований, так и в публикации их результатов. Опыт этот, безусловно, нужно нарабатывать, но вряд ли тренировочной площадкой следует признать лучшие научные журналы страны. Разумеется, если статья аспиранта соответствует их уровню — ее, очевидно, нужно в них публиковать, но при этом существует необходимость в наличии альтернативной — помимо журналов из "списка ВАК" — площадки для публикации работ СК и (в меньшей степени) СД.

Следует также указать на то, что обнародование и последующая оценка результатов диссертационных исследований заключается не только в публикации посвященных им статей в профильных рецензируемых журналах: СК (СД) должен выступить с докладом о своей работе на нескольких конференциях, а непосредственно перед защитой автореферат его диссертации публикуют в Интернете (для СД — на сайте ВАК, для СК — на сайте учреждения, при котором функционирует диссертационный совет) и отправляют в другие вузы по списку рассылки, текст диссертации предоставляют официальным оппонентам, которые должны дать объективную и непредвзятую оценку работе диссертанта. Здесь также существует комплекс проблем.

1. Оппоненты, хотя и считаются назначенными советом, как правило, являются хорошими знакомыми научного руководителя (консультанта) соискателя, и получение отзывов на диссертации, по сути дела, происходит на бартерной основе. В частности, хотя по положению ВАК защита диссертации разрешена даже в случае отрицательного отзыва одного из оппонентов и ведущей организации [3], обычно соискатели и их научные руководители стараются все организовать таким образом, чтобы все отзывы были положительными. Вряд ли такой подход можно считать объективной оценкой научных достижений соискателя.

2. На авторефераты, поступившие в другие вузы по списку рассылки, отзывы пишутся только в том случае, если адресатов об этом попросил научный руководитель соискателя. В противном случае конверты с авторефератами отправляются в мусорную корзину. Очевидно, что при таком подходе объективность полученных диссертационным советом (ДС) отзывов представляется сомнительной.

3. Будучи формально обязательным, требование о публикации автореферата на сайте ДС (призванное резко расширить потенциальную аудиторию читателей автореферата) соблюдается не все-

ми вузами. Кроме того, в отдельных случаях вуз ограничивается размещением объявления о предстоящей защите, не давая ссылки на автореферат и даже не предоставляя информации о дате защиты (что, по сути, превращает защиту из публичного мероприятия в закрытое и является прямым нарушением требований ВАК). Например, для очень многих объявленных защит на сайте Санкт-Петербургского государственного инженерно-экономического университета даты защит найти невозможно. Наконец, если авторефераты представленных к защите диссертаций на соискание ученой степени доктора наук найти сравнительно легко (они выложены на сайте ВАК), то в случае, если у ученого возникает желание ознакомиться с авторефератами кандидатских диссертаций, ему необходимо проверить сайты всех учреждений, при которых открыты ДС по соответствующей специальности, что зачастую затруднительно (кроме того, не все учреждения — в отличие от ВАК — поддерживают архив авторефератов).

4. Конференции, призванные обеспечить апробацию полученных исследователями результатов, за редким исключением собирают малое число участников, а сборники материалов публикуются тиражом, равным числу участников этой конференции + список обязательной рассылки. Найти эти материалы в вузовских библиотеках невозможно (за исключением библиотек тех вузов, где происходили данные конференции). Это также затрудняет ознакомление других исследователей с результатами соискателя. В ряде случаев докладчики ограничиваются заочным участием и просто присылают тезисы или материалы для публикации в сборнике трудов конференции. Вследствие этого никакой дискуссии по представленным в заочных докладах материалам не проводится, что превращает апробацию в фикцию.

В своем стремлении повысить качество диссертационных исследований ВАК активно обращается к потенциалу сетевого информационного пространства. Как уже упоминалось, есть вполне логичное в современных условиях требование о публикации авторефератов в Интернете, от рецензируемых журналов требуется наличие полнотекстовой версии в сети Интернет, и даже осуществляются Интернет-трансляции защит диссертаций (http://vak.ed.gov.ru/ru/news/anons/index.php?id4 = 901). Отметим, правда, что последняя мера пока не представляется эффективной — пропускная возможность Интернета у большинства людей, потенциально заинтересованных в такой трансляции, недостаточна для ее просмотра.

Как нам представляется, наращивание использования возможностей сетевого информационного пространства позволило бы решить многие из отмеченных выше проблем. Ниже предлагается несколько вариантов такого использования.

Электронный архив научных публикаций

Как уже отмечалось, необходимо, наряду с журналами из "списка ВАК", предложить СК и СД альтернативный канал для публикаций результатов их работ. Этот канал должен удовлетворять следующим основным требованиям:

- *доступность* публикаций для научной читательской аудитории — любой, желающий ознакомиться с публикациями того или иного соискателя, должен иметь возможность без проблем это сделать;
- *централизованность* — публикации диссертантов должны быть сгруппированы в одном месте для упрощения ознакомления с ними;
- *независимый отбор* — статьи, размещаемые в этом издании, должны проходить независимую экспертизу. В качестве критериев отбора должны выступать новизна полученных результатов и отсутствие плагиата, а также соответствие требованиям ВАК к оформлению публикаций в рецензируемых журналах (наличие ключевых слов, аннотации на русском и английском языках);
- *оперативность* — срок между подачей статьи и ее публикацией (в случае принятия) должен быть минимальным: не секрет, что подлинно интересные результаты у диссертантов появляются к концу срока обучения в аспирантуре. В традиционных журналах с их длительными сроками рассмотрения и публикации всегда есть опасность не успеть опубликоваться до защиты. В предлагаемом издании эта опасность должна быть устранена;
- *бесплатность* — статьи диссертантов должны быть опубликованы бесплатно (в том числе и без скрытых платежей наподобие платы за рецензирование, подписку или авторский экземпляр), причем речь идет о диссертантах всех категорий, а не только об аспирантах, как в нынешнем постановлении ВАК;
- *самостоятельность публикаций* — принимаются только статьи, написанные единолично диссертантом. С одной стороны, это позволит лучше учитывать вклад соискателя в написание своей диссертации (что не всегда просто, если в списке работ по теме диссертации есть только статьи, написанные в соавторстве, что не редкость в наше время). С другой стороны, это хотя бы отчасти поможет поставить барьер на пути порочной практики, когда список трудов научного руководителя раздувается за счет бегло просмотренных им статей, написанных его аспирантами, которые вынуждены указывать его в качестве соавтора.

Можно возразить, что в отдельных направлениях современной науки (в частности, в экспериментальной физике высоких энергий) сложилась такая практика, что в качестве авторов статей, рассказывающих о результатах проведенных экспериментов, указываются все люди, причастные к их проведению. Соответственно, требование указывать только одного автора будет нарушением авторских прав других участников исследования. Однако это возражение легко снимается: как правило, в экспериментальные команды входит и некоторое число ученых мирового уровня, которым не составляет труда опубликовать статью (с коллективным авторством, и в числе авторов будет указан и диссертант) в ведущих научных журналах мира. Сам же диссертант должен издать статью, повествующую о его личном вкладе — и такая статья вполне может иметь одного автора;

- *рейтинг статей* — необходимо, чтобы публикации в этом издании могли быть проранжированы по их популярности у читателей, так как это позволило бы дополнительно оценить важность излагаемых в них результатов. Разумеется, при этом нужно разработать строгий критерий ранжирования.

Легко убедиться, что этим требованиям удовлетворяет только публикация статей в специально предназначенном для этого сетевом издании. С учетом того, что такое издание должно удовлетворять потребности диссертантов всех специальностей, разумнее говорить не об электронном журнале, а об электронном архиве (или электронной библиотеке). Примерным аналогом такого архива можно было бы считать международный архив препринтов по физико-математическим специальностям arXiv (arxiv.org).

Создание предлагаемого архива могло бы происходить по следующей схеме.

1. Федеральное агентство по образованию (возможно, в партнерстве с ведущими вузами страны, прежде всего — МГУ и СПбГУ, так как именно эти вузы могут в будущем получить право выдавать собственные дипломы о высшем образовании и самостоятельно присуждать ученые степени) — далее будем называть их учредителями — разрабатывают концепцию такого архива (включая механизмы отбора статей, рейтингования и финансирования). В качестве соучредителей проекта могли бы также выступить РАН и государственные отраслевые академии, прежде всего — Российская академия образования.

2. Учредители заключают договор с кем-либо из крупнейших Интернет-компаний России о предоставлении дискового пространства для этого проекта. Идеальным кандидатом на роль партнера является, на наш взгляд, компания *Yandex*, так как это позволило бы существенно оптимизировать поиск по базе публикаций.

3. Запускается пробная версия архива. Вероятно, на этом этапе будут публиковаться только работы диссертантов из вузов-учредителей.

4. Если проект на практике подтверждает свою жизнеспособность, то запускается его рабочая версия, доступ к публикации в которой получают все соискатели, обучающиеся в российских вузах.

Процесс отбора и публикации статей мог бы происходить по следующему алгоритму.

1. На начальной стадии проекта в нем в качестве рецензентов регистрируются все ученые и преподаватели из вузов-участников, имеющие право руководить СК и выступать научными консультантами у СД. Каждый из этих специалистов в обязательном порядке указывает свой часто проверяемый электронный почтовый ящик и получает уникальные логин и пароль для входа в систему рецензирования архива. Каждый из этих специалистов, далее, должен указать коды специальностей (по номенклатуре ВАК), по которым он имеет право выступать руководителем и/или консультантом, а также код специальности, по которой он защищал свою докторскую и/или кандидатскую диссертацию. Возможно, на следующем этапе в качестве таких рецензентов могут быть зарегистрированы бывшие СК и СД, успешно прошедшие защиту своих диссертаций, оставшиеся работать в системе науки и высшего образования, и чьи публикации в архиве отличаются высоким рейтингом.

2. Аналогичным образом в системе регистрируются все СК и СД, проходящие в настоящее время обучение. Помимо своих специальностей, при регистрации они также указывают коды других специальностей, являющихся смежными или представляющими для них интерес (пример — основная специальность 08.00.13 "Математические и инструментальные методы экономики", дополнительные специальности — 05.13.18 "Математическое моделирование" и 01.01.09 "Дискретная математика и математическая кибернетика").

3. Таким же образом в системе регистрируются руководители и ученые секретари всех диссертационных советов, а также заведующие отделами аспирантуры и докторантуры — для получения отчетов о публикациях прикрепленных к ним СК и СД.

4. На сайте архива публикуются требования к оформлению работ, размещаемых в архиве.

5. Диссертант, желающий опубликовать свою работу, приводит ее в соответствие с требованиями к оформлению и отсылает ее в архив через специальную контактную форму, размещенную на его сайте. В этой форме обязательно указываются ключевые слова, название вуза, к которому при-

креплен соискатель, ФИО его научного руководителя (консультанта) и код специальности, список статей, уже опубликованных в электронном архиве, на которые он ссылается в своей работе. Возможно, во избежание публикации статей третьими лицами (как это практикуется при написании диссертаций "на заказ"), отправка статей может быть осуществлена только с компьютера вуза, к которому прикреплен диссертант.

6. Система подтверждает диссертанту получение его статьи и проводит проверку присланного текста на плагиат. Некоторые российские издания уже проводят такую автоматизированную проверку с помощью системы "Антиплагиат", например "Вестник Российской академии государственной службы" (http://oad.rags.ru/vestnikrags/index.htm). Этот пункт предполагает интеграцию системы проверки на плагиат с системой электронного архива.

7. Если в тексте был выявлен плагиат, диссертанту сразу же отправляется письмо с уведомлением об отказе в публикации и указанием причины. Аналогичное письмо отправляется его научному руководителю и в соответствующий диссертационный совет. Данное письмо должно быть включено в личное дело диссертанта. В случае повторной присылки статьи с элементами плагиата диссертант исключается из аспирантуры/докторантуры. При этом необходимо разработать также процедуру апелляции — нельзя исключать возможность машинных ошибок.

8. Если в тексте плагиата выявлено не было, статья отправляется по электронной почте трем случайным рецензентам из числа зарегистрированных научных руководителей/консультантов, чьи коды специальностей соответствуют коду специальности статьи. Число рецензентов должно быть разным для статей, написанных СК и СД. При этом рецензенту не сообщается, на получение какой именно степени (докторской или кандидатской) претендует автор статьи. Среди этих рецензентов, разумеется, не должно быть научного руководителя автора. Рецензирование анонимное — автору и рецензентам не сообщаются ФИО и названия вузов друг друга. Вероятно имеет смысл предусмотреть возможность указания автором при отправке статьи ФИО рецензентов, к которым статью направлять не следует (необходимо для того, чтобы представители соперничающих научных школ не могли блокировать публикации противоположной стороны). Рецензентам могут быть присвоены разные веса в зависимости от их ученой степени и звания, членства в академиях наук, рейтинга вуза или НИИ, в которых они работают и т. д.

9. Рецензенты должны в течение двух недель оценить соответствие статьи определенным критериям (полнота изложения материала, стиль изложения, наличие научной новизны и т. д.) по десятибалльной шкале (оценивается не работа в целом, а каждый критерий в отдельности). Для выставления оценок на сайте электронной библиотеки предусматривается специальная форма. В этой форме также может быть специальное поле, в котором рецензент по желанию может указать наиболее существенные недочеты статьи.

10. После поступления оценок работы в систему электронных публикаций в ней автоматически выставляется итоговая оценка статьи по следующей формуле:

$$ИОС = \frac{\sum\limits_{j=1}^{m} \frac{W_j \sum\limits_{i=1}^{n} w_i Q_{ij}}{\sum\limits_{i=1}^{n} w_i}}{\sum\limits_{j=1}^{m} W_j},$$

где $ИОС$ — итоговая оценка статьи; n — число критериев, по которым оценивается статья; m — число рецензентов, которым статья была отправлена на рассмотрение; w_i — вес i-го критерия в совокупной оценке; W_j — вес j-го рецензента; Q_{ij} — оценка соответствия статьи i-му критерию, выставленная j-м рецензентом.

11. Тексты рецензий сохраняются в специальном закрытом отделе электронного архива и могут быть предоставлены по запросу ВАК.

12. Статья принимается к публикации в системе, если ее $ИОС$ больше или равна некоторому минимальному значению. Это минимальное значение должно быть разным для "докторских" и "кандидатских" статей (для статей, написанных СД, $ИОС_{\min}$ должна быть выше).

13. К одному и тому же рецензенту нельзя обращаться более 5 раз в течение календарного года. При этом он обязан отрецензировать не менее трех присланных ему статей. Если он отрецензировал менее трех статей (из присланных пяти), статьи руководимых им СК и СД к публикации в архиве приниматься не будут.

14. В случае необходимости статья отправляется диссертанту на доработку. В течение недели он обязан либо прислать исправленную статью, либо отказаться от ее публикации. После этого она отправляется на повторное рассмотрение одному из рецензентов. В случае его одобрения (оценка по десятибалльной шкале от 5 и выше без повторной подробной рецензии) статья немедленно публикуется в архиве.

15. После публикации статьи информацию о ее размещении получают все соискатели, указавшие код соответствующей специальности в списке своих основной и дополнительной специальности. Такую же информацию получат все научные руководители по этой специальности.

16. На странице архива, на которой публикуется статья, размещается форма для отзывов. Эти отзывы (если они поступят) в последующем можно будет прочесть под текстом самой статьи. При этом будут существовать строгие правила составления текстов отзывов. За соблюдением правил будут следить модераторы (возможно, текст отзыва будет публиковаться только после его предварительного прочтения модератором).

17. Лица, желающие разместить свой отзыв о статье, должны будут указать в соответствующей форме код своей специальности, название вуза и ФИО. Это повысит ответственность автора отзыва и вынудит его с особым тщанием отнестись к подготовке текста.

18. Автор статьи получает на свой электронный почтовый ящик уведомление о поступившем отзыве и, при желании, может вступить с автором отзыва в дискуссию. При защите диссертации распечатки таких дискуссий могут быть приложены к личному делу СК (СД), как подтверждение апробации его результатов. Как представляется, такая форма апробации является гораздо более эффективной, чем научные конференции.

19. Лица, допустившие неэтичные формулировки в текстах отзывов, на определенный срок лишаются права публиковать свои статьи в электронном архиве.

20. Система обеспечивает поиск публикуемых в ней работ по коду специальности, ФИО автора, ФИО научного руководителя, вузу и ключевым словам.

21. Перед защитой соискатель по специальному запросу получает от архива индекс цитирования своих статей (внутри системы) и информацию о числе отзывов о них. Эти величины должны быть выше некоторого определенного порога (разного для разных специальностей — так как у разных научных направлений разная популярность, и статьи по ним будут интересны разному числу исследователей; соответственно число отзывов будет различаться). Введение представления о разных численных критериях для разных специальностей представляется исключительно важным, так как именно отсутствие таких различий является одним из основных поводов для критики наиболее известного в настоящее время индекса цитирования, рассчитываемого Институтом научной информации (г. Чикаго, США). Если число отзывов ниже этого порогового значения, то соискатель не допускается до защиты, так как это означает, что результаты его исследований не вызвали интереса у коллег, а сама работа не соответствует критерию актуальности. Расчет этих пороговых значений может проводиться по следующей формуле:

$$Q_{\min} = a_0 N_{\text{spec}} + b_0 n_{\text{spec}} + \sum_{i=1}^{m} (a_i N_i + b_i n_i),$$

$Q_{\min}$ — минимальное требуемое число отзывов на все статьи, опубликованные СК или СД в электронном архиве; N_{spec} — среднее число СК и СД данной специальности по стране за нормативный срок обучения в аспирантуре (докторантуре); n_{spec} — среднее число научных руководителей данной специальности по стране за нормативный срок обучения в аспирантуре (докторантуре); N_i — среднее число СК и СД смежных специальностей по стране за нормативный срок обучения в аспирантуре (докторантуре); n_i — среднее число научных руководителей по смежным специальностям по стране за нормативный срок обучения в аспирантуре (докторантуре); m — число смежных специальностей; a_0, b_0, a_i, b_i — поправочные множители. Они могут быть как одинаковыми для всех специальностей, так и различаться в зависимости от специальности, формы и срока обучения.

По аналогичной формуле можно рассчитывать минимально необходимое число ссылок на статьи диссертанта, с той разницей, что в этом случае из формулы будут исключены члены, учитывающие число научных руководителей (так как в архиве публикуются только статьи диссертантов, и число цитирований учитывается только в рамках системы).

22. Во избежание чрезмерных затрат на поддержание архива возможно имело бы смысл ограничить объем одной публикации и число публикаций за время обучения в аспирантуре/докторантуре. На наш взгляд, для СК может быть достаточно пяти публикаций, для СД — пятнадцати. При этом СК обязан опубликовать не менее трех статей, а СД — не менее десяти. Публикация большего числа статей (сверх 5 и 15 соответственно) возможна при условии оплаты. Взимание этой платы нарушением требования бесплатности не будет, так как диссертанту предоставляется право бесплатной публикации определенного числа статей.

Предложенная схема способна обеспечить высокую прозрачность процесса подготовки специалистов высшей квалификации и снизить число так называемых "заказных" диссертаций.

Финансирование проекта может осуществляться за счет:

- бюджетных субсидий;

- отчислений от вузов, имеющих в своем составе аспирантуру и докторантуру; размер отчислений зависит от числа фактически обучающихся СК и СД;
- добровольных пожертвований;
- грантов;
- платы за публикацию статей сверх предусмотренной квоты;
- размещения рекламы (при этом размер рекламных материалов не должен превышать определенной части от размера страницы, и рекламные материалы не должны мешать знакомиться с научными публикациями — в частности, не быть "всплывающими"). Партнерство с компанией *Yandex* в этом случае было бы идеальным, так как она могла бы предложить контекстную рекламу в рамках поиска по электронному архиву (ссылки на магазины, где можно было бы купить соответствующую научную литературу, платные курсы и т. д.).

Отдельным вопросом является оплата труда рецензентов. На наш взгляд, рецензирование должно проводиться бесплатно, и для этого есть три причины.

1. В большинстве ведущих научных журналов мира рецензирование проводится бесплатно (хотя и в гораздо более длительные сроки, чем предлагается в разрабатываемом проекте электронного архива).

2. По сути дела, научные руководители СК и СД в рамках данного архива продолжат практику "бартерного обмена" рецензиями, но на качественно новой (анонимной и объективной) основе — сегодня научный руководитель пишет отзыв на статью некоего неизвестного ему аспиранта, а завтра отзыв на статью его аспиранта напишет неизвестный ему другой научный руководитель. Так как в настоящее время такой обмен рецензиями (правда, на диссертации) осуществляется бесплатно, на дружеской основе, то и в рамках данного проекта требовать (и предлагать) оплату за рецензирование представляется нелогичным. Скорее, бесплатное рецензирование будет являться еще одним проявлением солидарности научного сообщества.

3. Для экономии времени и труда рецензентов отзывы они должны будут представлять по специальным упрощенным формам. Повторное же рецензирование (после доработки статьи автором) будет, как уже говорилось выше, представлять собой не собственно написание рецензии на рукопись, а оценку ее по десятибалльной шкале.

Очевидно данный электронный архив может быть использован только для публикации результатов "открытых" диссертаций. Однако диссертации с грифом "для служебного пользования" и не предназначены для оценки их широкой научной общественностью, и потом для авторов таких диссертаций необходимость в публикации своих статей в архиве отсутствует (при этом, разумеется, ничто не препятствует публикации в электронном архиве открытой части их исследований).

Рецензируемые журналы

При учете публикаций в журналах из "списка ВАК" возникают следующие проблемы:

- "список ВАК" был впервые введен только в 2002 г., и неясно, как расценивать публикации в журналах, которые входили во все редакции "списка ВАК" (например, различные серии "Вестников" СПбГУ и МГУ и большинство журналов РАН), но были сделаны до 2002 г.;
- "список ВАК" регулярно пересматривается, часть журналов из новых редакций исключается, и в него включаются новые журналы. При этом отсутствует однозначный алгоритм того, как засчитывать статьи, опубликованные в выбывших из этого списка журналах. Дело в том, что решение о публикации статьи принимается редакцией журнала задолго до ее фактической публикации. Соответственно, решение о принятии статьи к печати может быть принято до того, как журнал был исключен из этого списка (т. е. формально на момент одобрения статьи редакцией журнал соответствовал требованиям ВАК, а статья, таким образом, соответствовала уровню ведущего научного журнала и, разумеется, с точки зрения логики, не перестала соответствовать этому уровню и после исключения журнала из списка). Аналогичная проблема возникает с журналами, только что включенными в "список ВАК" (с той разницей, что статья, принятая к печати до включения в этот список, но опубликованная после занесения журнала в "список ВАК", формально уровню ведущего научного журнала не соответствует);
- практическая проблема заключается в том, что диссертационный совет (ДС), принимая диссертацию к защите, оказывается вынужден сверять названия журналов, в которых были опубликованы статьи соискателя, со всеми редакциями "списка ВАК" (потому что часть этих журналов могла в отдельных редакциях списка отсутствовать), а затем проверять, совпадает ли дата публикации статьи с периодом пребывания соответствующего журнала в "списке ВАК". Желая избавиться от этой работы, ряд ДС признают публикации только в тех журналах, которые фигурируют в "списке ВАК" на момент представления диссертации к защите (независимо от того, были ли эти журналы в "списке ВАК" на момент публикации статьи), и

не признают статьи в остальных журналах (даже если на момент публикации статьи эти журналы присутствовали в "списке ВАК"), Это в определенной степени обессмысливает "список ВАК". Кроме того, легко представить ситуацию, в которой аспирант с одной "ваковской" статьей при использовании этой модели может оказаться без "ваковских" публикаций – если пересмотр списка случился перед самой защитой. Очевидно, что такая ситуация неприемлема.

Первая из перечисленных выше проблем может быть устранена только путем принятия соответствующего постановления ВАК, регламентирующего порядок учета ранних (до 2002 г.) публикаций (что представляет собой довольно сложную бюрократическую проблему и, очевидно, находится вне предмета данной статьи). Однако решением двух последних проблем может стать размещение на сайте ВАК общедоступной базы всех журналов, когда-либо входивших в "список ВАК" (с зеркалами на других сайтах, например, на сайте Научной электронной библиотеки elibrary.ru, а также на сайте предлагаемого электронного архива). Эта база должна включать в себя следующую информацию:

1) название журнала и список всех его прошлых названий;

2) перечень экспертных советов, рекомендовавших этот журнал;

3) название издательства;

4) контактные координаты редакции журнала (или всех редакций, если журнал выходит в нескольких сериях, каждая из которых готовится отдельной редакцией) — почтовый адрес, адрес для посетителей, телефон, факс, электронная почта, ФИО лиц, которым должна быть адресована рукопись;

5) периодичность выхода;

6) средние сроки рассмотрения и публикации статей (информация, крайне важная для соискателей кандидатской степени, так как многие из них публикуются в самый последний момент; сведения о сроках рассмотрения материалов позволили бы им рационально планировать график публикаций);

7) перечень номеров журнала, которые засчитываются ВАК. Необходимо указывать именно перечень номеров, а не даты пребывания журнала в "списке ВАК", так как публикации следует засчитывать только после истечения определенного срока с момента включения журнала в этот список, а переставать засчитывать — по истечении определенного срока с даты исключения из списка. Этот срок должен быть равен среднему сроку рассмотрения представленной статьи в соответствующем журнале.

База данных должна допускать поиск по названиям журналов и по рекомендовавшим эти журналы экспертным советам. Поиск должен быть возможен по всем названиям, которые носил какой-либо журнал.

Предлагаемая база смогла бы существенно упростить и сделать более прозрачным процесс публикации результатов диссертационных исследований.

Разумно было бы также создать аналогичную информационную базу по диссертационным советам с указанием их шифра, состава, сроков действия, перечня специальностей и отраслей науки, по которым они могут проводить защиты, а также контактных координат.

Заключение

Более широкое использование возможностей Интернета с присущими ему сравнительно низкими издержками на публикацию научных работ, обеспечением свободного доступа к опубликованным материалам и возможностью открытого обсуждения достигнутых результатов позволило бы существенно повысить прозрачность процедуры подготовки диссертации и поставить барьер на пути написания диссертации "на заказ".

Предлагаемый электронный архив статей СД и СК мог бы стать эффективным и общедоступным альтернативным каналом публикации результатов диссертационных исследований (в том числе и благодаря предусмотренной в нем системе оповещения о новых статьях и модуле проверки на плагиат). В случае же введения обязательной публикации на сайте архива авторефератов диссертаций (с предоставлением отзывов на них также через систему архива в электронном виде), размещения материалов международных и всероссийских конференций, а также информации о журналах из "списка ВАК" этот электронный архив может стать основой для формирования информационного подпространства поддержки диссертационных исследований, выполняющего следующие функции:

- *контрольная* — обеспечение высокого качества диссертационных работ (путем проверки на плагиат, рецензирования связанных с ними статей и обсуждения этих статей научным сообществом);
- *инновационная* — канал для публикации статей, материалов конференций и авторефератов;
- *информационная* — публикация сведений о рецензируемых журналах, перечня диссертационных советов с их контактными координатами, пособия по подготовке диссертационных

работ, перечня вопросов экзаменов кандидатского минимума и открытых пособий по подготовке к ним и т. д.;

- *социальная* — взаимодействие между членами научного сообщества (то есть создание на основе электронного архива научной социальной сети).

Для начала каждый диссертант мог бы иметь возможность создать в системе свою собственную страничку с личной и рабочей информацией о себе.

Потребность в таком информационном подпространстве настолько велика, что научное сообщество стихийно пытается формировать его (в качестве примера можно привести портал Phido.ru, для которого автор данной статьи подготовил перечень контактных координат журналов из "списка ВАК"). Тем не менее, без наличия государственной поддержки такие порталы будут всего лишь паллиативом и не смогут эффективно справляться с выполнением контрольной и информационной функций.

Список литературы

1. **Информационное** сообщение о порядке формирования Перечня ведущих рецензируемых научных журналов и изданий, в которых должны быть опубликованы основные научные результаты диссертаций на соискание ученой степени доктора и кандидата наук (от 14.10.2008). URL: http://vak.ed.gov.ru/ru/list/infletter-14-10-2008.
2. **Перечень** ведущих рецензируемых научных журналов и изданий, в которых должны быть опубликованы основные научные результаты диссертации на соискание ученой степени доктора и кандидата наук (редакция апрель 2008 года). URL: http://vak.ed.gov.m/common/img/uploaded/VAK/files_help_desk/per-04-2008.doc.
3. **Положение** о порядке присуждения ученых степеней. Утверждено постановлением Правительства Российской Федерации № 74 от 30.01.2002.
URL: http://vak.ed.gov.ru/ru/docs/ ?id4 = 212&i4 = 24.
4. **Решение** президиума об исключении журнала из Перечня изданий (от 17.10.2008).
URL: http://vak.ed.gov.ru/ru/news/ allnews/index.php?id4 = 1163.
5. **Российский** научный журнал поймали на публикации заведомой ерунды // Грани.ру. 30.09.2008.
URL: http://grani.ru/Society/Science/m.142082.html.

Информация

07—12 сентября 2009 г.
в Севастополе, Украина, состоится
Международная научно-техническая конференция

Информационные технологии и информационная безопасность в науке, технике и образовании "ИНФОТЕХ-2009"

Организаторы конференции:

- Министерство образования и науки Украины
- Севастопольский национальный технический университет, г. Севастополь, Украина
- Санкт-Петербургская государственная академия аэрокосмического приборостроения, г. Санкт-Петербург, Россия
- Институт проблем информатики РАН, г. Москва, Россия
- Технический университет г. Люблин, Польша

Председатель: д-р техн. наук, проф., ректор СевНТУ **Пашков Е. В.** (г. Севастополь, Украина)

Сопредседатели:
Скатков А. В., д-р техн. наук, проф., зав. каф. СевНТУ, г. Севастополь, Украина;
Гуревич И. М., канд. техн. наук, ст. науч. сотр. ИПИ РАН, г. Москва, Россия;
Котов В. М., д-р физ.-мат. наук, проф., зав. каф. БГУ, г. Минск, Белоруссия.

Тематические направления конференции

- Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей
- Системный анализ, управление и моделирование
- Системы и средства искусственного интеллекта
- Методы и системы защиты информации, информационная безопасность
- Компьютерные системы, сети и компоненты
- Автоматизированные системы управления и информационные технологии

Планируется проведение круглых столов по основным направлениям работы конференции. Предложения по тематике беседы принимаются на официальном сайте конференции и в ходе ее работы.

Предусматривается культурная программа: экскурсии по историческим местам г. Севастополя, экскурсия в океанариум НАН Украины, морская прогулка, посещение Херсонеса, Балаклавы.

Подробную информацию о конференции см. на официальном сайте
http://www.sev-infotech.info/

CONTENTS

depends on part of lifetime, in which this transistor is under stress, in other words, on stress probability. In this paper, we propose the correct algorithm of calculating stress probability for every pMOS transistor of complex CMOS gate. Comparing to simple "naive" approach, our algorithm takes into account two additional factors: correlations between signals at gate inputs, and VDD-potential coming through "bottom" of pMOS transistor. Numerical experiments show the importance of accounting for both these factors.

Keywords: analysis of VLSI circuits, CMOS gate, negative bias temperature instability (NBTI).

In this work the new method for early design-stage estimate of the field-programmable gate array(FPGA)-based digital devices' characteristics is proposed. For the purpose of this article early design-stages include research and problem-solving stages(R&PS). The requirements for the new method are specified. It is shown that the proposed method utilizing two levels of macromodels satisfies the specified requirements. This method enables the exploration of greater amount of design variants thus shrinking the R&PS time and improving the project's quality.

Keywords: macromodel, system-level design, FPGA.

In given article are considered model of the text of any maintenance which allows displaying semantic coherence and sequence of the text in the form of structure with use of logic scales. And also allows defining visually keywords with a context, to organize analyzed texts in uniform, complete information structure, providing the subsequent analysis of set of documents of allocation from them the general thematic clusters.

Keywords: word-combination, the analysis, maintenance, search, context, classification.

The article is dedicated to the analysis of achievements in computer knowledge processing from natural language text. Present-day applied research trends devoted to knowledge mining and knowledge processing of Internet textual data are represented. An experimental system for evaluation of consumer characteristics of products is described, which is based on the analysis of consumer opinions placed in a social network on the Internet.

Keywords: knowledge extraction, knowledge processing, automatic knowledge management systems, text mining, social networks on the Internet.

In this paper some practical aspects of a large-scale intellectual information processing system development are considered. The system under consideration is designed for information searching and thematic content filtering in large data sets like enterprise-wide text collections, or Internet.

Keywords: collections of documents, automatic processing, significance, information, search engine, text indexing.

In this paper the new approaches uncovering possibilities of application of the R-functions theory to construction of the normalized equations of complex geometric objects possessing a symmetry from various applied areas are offered.

Keywords: geometric object, symmetry, translational element, coordinates transformation, R-functions.

The article presents a method for reconstruction 3D scenes of urban environment from noncalibrated sequence of pictures. The proposed approach is based on modeling of polygonal objects from the spatial lines derived from vectorized source images, their calibration and matching lines algorithm on the images. This method uses orthogonality and parallelism of edges and faces in objects scenes.
Keywords: 3D reconstruction, urban scene, epiline, epipolar geometry, vectorization, matching.

The present article contains a description of a method of control of the quality of results obtained in doctoral and post-doctoral theses. This method is based on a specialized electronic archive of publications of Ph. D. students. A list of requirements this archive should meet is proposed. An algorithm of this archive is described. Financial model of this archive is proposed.
Keywords: results of doctoral research, electronic archive, approbation.

Адрес редакции:

107076, Москва, Стромынский пер., 4

Телефон редакции журнала **(499) 269-5510**
E-mail: it@novtex.ru

Дизайнер *Т.Н. Погорелова*. Технический редактор *О. А. Ефремова*.
Корректор *Е. В. Комиссарова*
Сдано в набор 07.05.2009. Подписано в печать 18.06.2009. Формат 60×88 1/8. Бумага офсетная. Печать офсетная.
Усл. печ. л. 9,8. Уч.-изд. л. 10,80. Заказ 553. Цена договорная.
Журнал зарегистрирован в Министерстве Российской Федерации по делам печати,
телерадиовещания и средств массовых коммуникаций.
Свидетельство о регистрации ПИ № 77-15565 от 02 июня 2003 г.
Отпечатано в ООО "Подольская Периодика"
142110, Московская обл., г. Подольск, ул. Кирова, 15