УДК 681.322

С.В. Ленков,
доктор технических наук, профессор,
Ю.А. Гунченко,
доктор технических наук, доцент,
В.В. Савенчук

ПОСТРОЕНИЕ И АНАЛИЗ ПОМЕХОЗАЩИЩЕННЫХ ПРОЦЕССОРНЫХ АРХИТЕКТУР

В работе проанализированы принципы построения помехозащищенных архитектур процессоров для современных вычислительных систем. Описаны функционирование и особенности модернизации для трех архитектур, показаны их преимущества и недостатки, приведены зависимости скорости работы (количества исполняемых инструкций) от вероятности ошибки вычислений.

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

У роботі проаналізовано принципи побудови завадостійких видів архітектур процесорів для сучасних обчислювальних систем. Описано функціонування і особливості модернізації для трьох видів архітектури, показано їх переваги та недоліки, наведено залежності швидкості роботи (кількості інструкцій, що виконуються) від імовірності помилки обчислювань.

Ключові слова: *процесор, завадостійка архітектура, сигнатура, обчислювальна система.*

In this paper the principles of jam protected CPU design for modern computing systems are analyzed. Functioning and features of modernization for three CPU designs are described, their advantages and disadvantages are shown, dependences of work's speed (number of executed instructions) from the probability of an error of calculations are given.

Keywords: *processor, CPU design, signature, computing system.*

Введение

В современных вычислительных системах специального назначения часто остро стоит проблема надежности. При этом решаются три типа задач:

– повышение надежности хранения данных;

– повышение надежности передачи данных;

– повышение надежности вычислений (достоверности полученных данных).

Задачи первого и второго типов направлены на обнаружение и устранение ошибок, возникающих при передаче или хранении данных, и основы их решения

имеют математический либо физический характер. Существует множество различных способов повышения надежности хранения и передачи данных: использование контроля (бита) четности, кодов Хемминга, Хаффмана и других, более сложных кодов.

Решение же задач третьего типа связано с изменениями архитектуры вычислительных устройств и самой системы в целом.

Постановка задачи

Необходимость повышения надежности вычислений наиболее сильно ощущается при организации распределенных вычислений, а также построении распределенных систем управления разнообразного назначения. Это стало возможным благодаря достаточному уровню развития современных компьютерных сетей, позволяющих использовать большое количество отдаленных компьютеров. Существующие подходы к созданию систем распределенной обработки данных в сетевых ресурсах строятся как технологические надстройки над стандартными операционными системами. В результате получаются многослойные программные комплексы, управление которыми требует сложного администрирования.

Примеры специализированных систем, которые дают представления о масштабах и сферах применения распределенных систем обработки данных, приведены в таблице 1.

Таблица 1

Примеры распределенных систем

Название	Цель	Стоимость
LHC (Большой Адронный Коллайдер), CERN	Обработка 1500 Тбайт ежегодно на 10 тыс. компьютерах, связанных сетями	$1 млрд.
«Связьинвест», Россия	Обслуживание 40 млн. абонентов проводных сетей связи	$480 млн.
Future Combat System, US Army	Соеденить всех участников боевых действий и военную технику в единую командную сеть	$177 млрд.

Цель работы

Таким образом, проблема повышения надежности вычислений на данный момент является актуальной, а ее решения — востребованы.

Повышение надежности вычислений при помощи изменения архитектуры может быть как косвенным (например, при построении систолических вычислительных систем, когда каждый вычислительный элемент решает однотипные задачи, что позволяет упростить его структуру, а в более простой структуре легче достигается необходимая надежность), так и напрямую влиять на надежность вычислений. Рассмотрим пример такой системы – структуру процессора [2; 3].

Основной материал

Архитектура 1. На рис. 1 представлена структурная схема помехозащищенного процессора. Блок выполнения команд 101 соединен со счетчиком команд 102, файл-регистром 103, буфером записи 104, блоком генерации сигнатуры 105 и

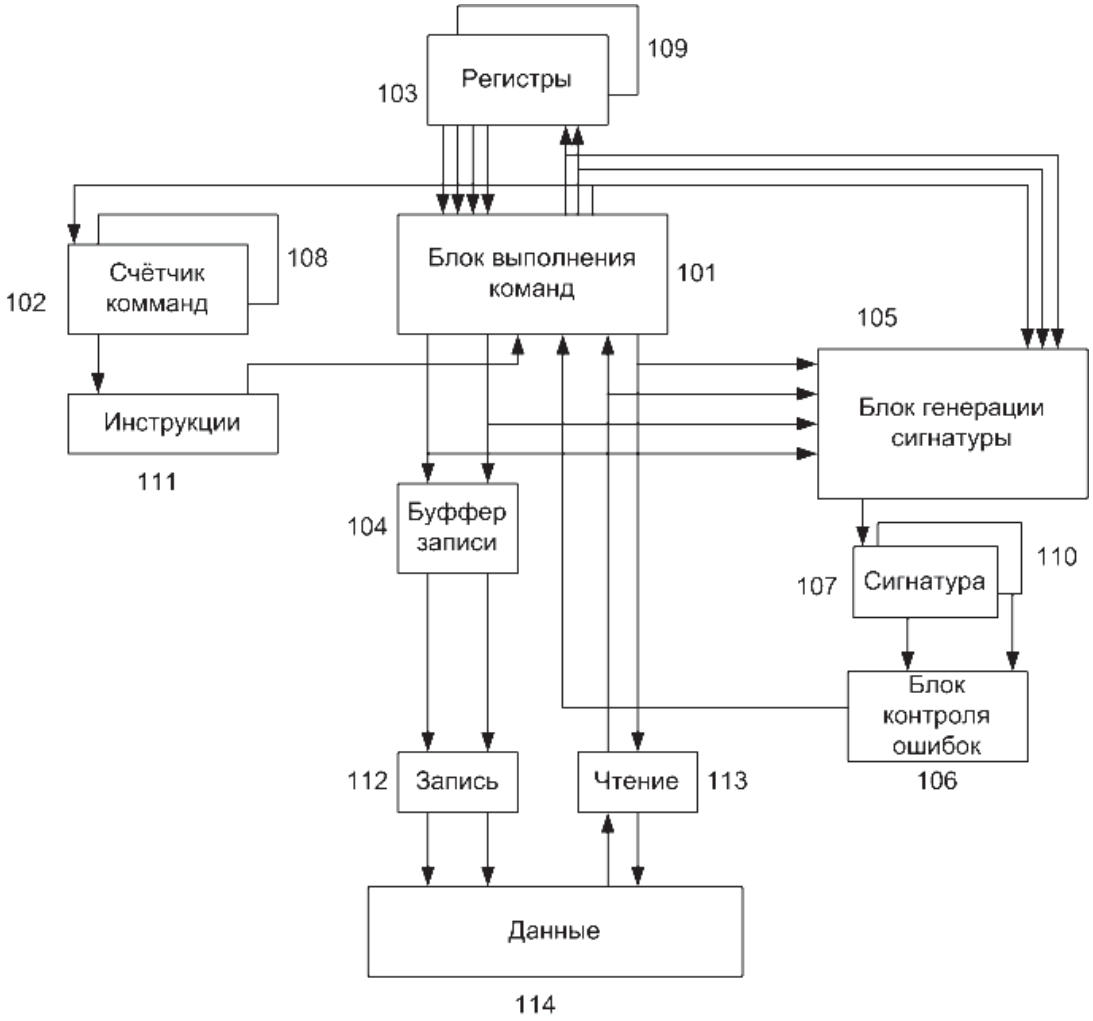

Рис. 1. Структура процессора

блоком контроля ошибок 106. Счетчик команд 102, файл-регистр 103 и регистр сигнатуры 107 соединены с соответствующими контрольными регистрами 108, 109, 110.

Файл-регистр 103 имеет контрольный регистр 109, который может сохранять текущее состояние файл-регистра (создание контрольной точки) и может восстанавливать значение файл-регистра из контрольной точки.

Счетчик команд 102 имеет контрольный регистр 108, способный сохранять текущие значения счетчика команд и восстанавливать его значение.

Блок выполнения команд 101 выполняет инструкции, содержащиеся в памяти 111, на которые указывает счетчик команд 102. Инструкции считываются и заставляют блок выполнения команд 101 оперировать содержимым файл-регистра 103. Блок выполнения команд 101 также может сохранять данные 112 и загружать 113 из памяти 114.

Буфер записи 104 содержит данные, которые должны быть записаны в память. Эта операция выполняет две функции. Во-первых, буфер записи 104 освобождает процессор от остановок во время операций записи. Во-вторых, буфер записи 104 содержит данные, которые могут быть использованы или отвергнуты в случае возникновения ошибки. Заметим, что буфер записи 104 не может быть источником данных, если адрес загружаемых данных совпадает с адресом данных, записанных в буфере записи 104. Это ограничение необходимо для достижения отказоустойчивости данной архитектуры.

Блок генерации сигнатур 105 принимает все результаты, вычисленные блоком выполнения команд 101. К ним также относятся все загруженные данные. Способ вычисления сигнатуры в данной работе не рассматривается. Блок генерации сигнатур 105 сохраняет вычисленную сигнатуру в соответствующий регистр 107. Регистр сигнатуры 107 имеет свой контрольный регистр 110, в который записывается текущее значение сигнатуры, при этом контрольная копия сигнатуры 110 никогда не записывается обратно в регистр сигнатур 107.

Блок контроля ошибок 106 контролирует запись в контрольные регистры 108, 109, 110, контролирует очистку буфера записи 104 и сравнивает текущую сигнатуру 107 с предыдущей 110. Псевдокод для блока контроля ошибок 106 представлен в виде блок-схем на рис 2, 3 и 4.

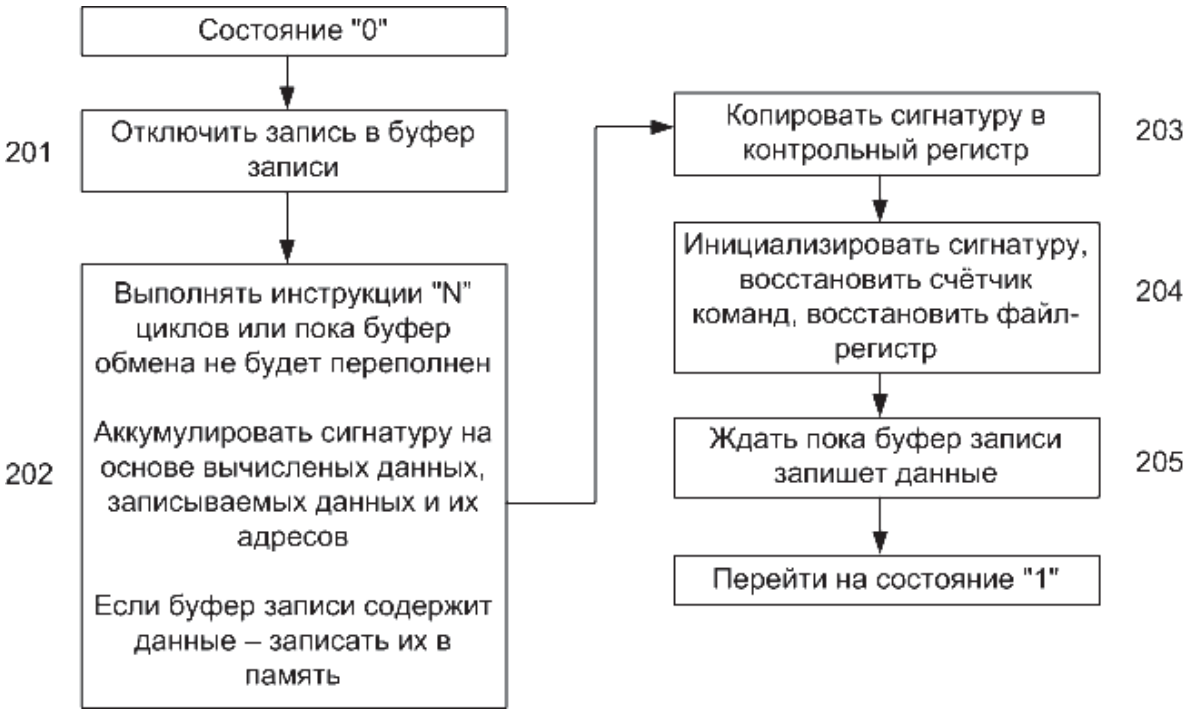

Рис. 2. Состояние “0”

В состоянии “0” процессор выполняет инструкции для формирования сигнатуры, которая будет использоваться в следующем состоянии для обнаружения ошибки.

Ссылаясь на рисунки 1 и 2, в начале состояния “0” отключается запись в буфер записи 104 (блок 201). Буфер записи 104 мог содержать данные, полученные во время предыдущих команд. Во время состояния “0” операции записи ставятся

в очередь записи в буфер 104. Эти операции не проходят никакой обработки после подсчета числа таких операций и сохранения адресов и данных в блок генерации сигнатуры 105.

Во время выполнения блока 202 инструкции выполняются до тех пор, пока не будет выполнено одно из двух условий. Выполнение команд прекращается в случае:

– если количество инструкций равно заданной постоянной “N”;

– если счетчик в буфере записи 104 указывает, что число операций записи в буфер записи 104 равно его размеру.

За это время сигнатура или несколько сигнатур аккумулируются на основе вычисленных результатов, сохраняемых данных и их адресов в регистре сигнатуры 107. Во время выполнения блока 202, если какая-либо информация хранилась в буфере записи 104, она будет записана в память одновременно с выполнением инструкций.

После завершения предыдущего блока на достижении одного из двух условий, описанных выше, накопленная сигнатура копируется из блока сигнатуры 107 в контрольный регистр 110 (блок 203).

В блоке 204 регистр сигнатуры 107 инициализируется, счетчик команд 102 восстанавливается из контрольного регистра 108, и файл-регистр 103 восстанавливается из контрольного регистра 109. До перехода в состояние “1” блок 205 завершает выполнение всех операций записи из буфера записи 104.

Рисунок 3 демонстрирует состояние “1”.

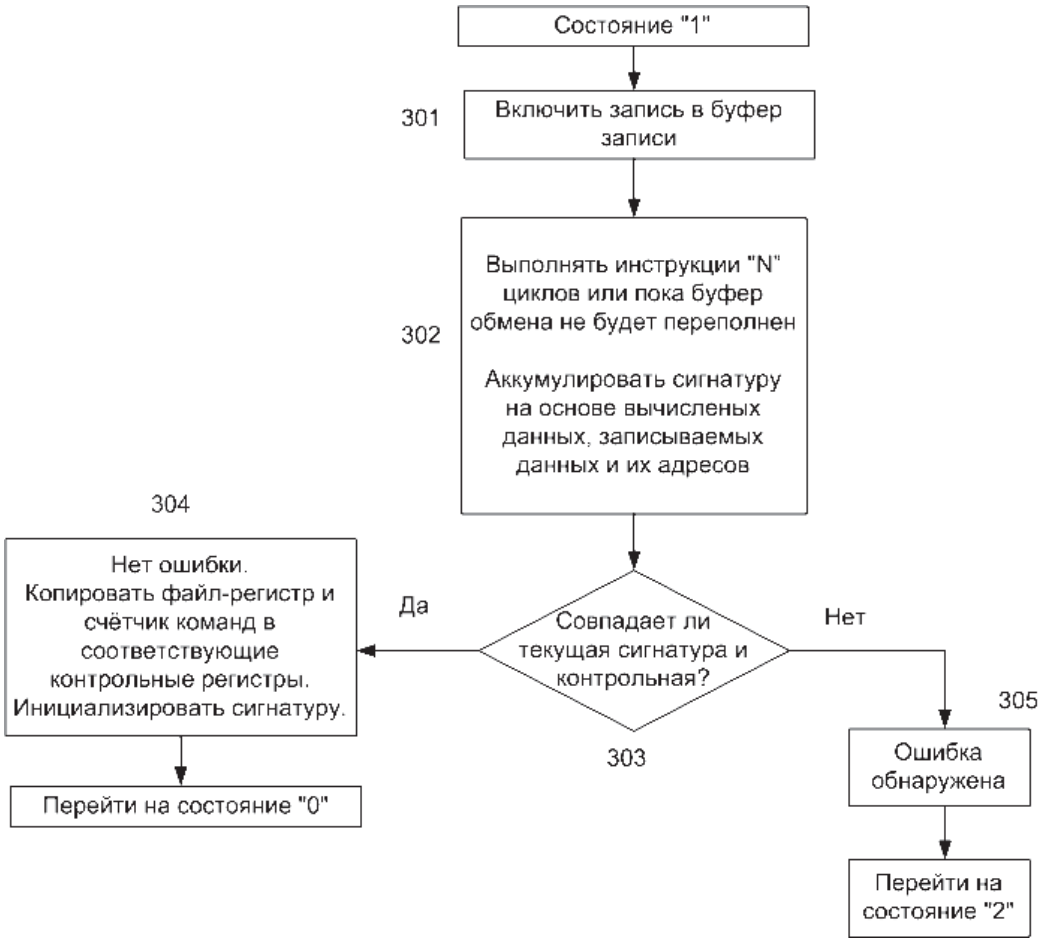

Рис. 3. Состояние “1”

В этом состоянии заново выполняются все команды для определения наличия ошибки. В начале состояния “1” блок 301 разрешает запись в буфер записи 104. Блок 302 аналогичен блоку 202. Таким образом, выполнение инструкций прекратится при достижении одного из двух условий:

– если количество тактов равно заданной постоянной “N”;

– если счетчик в буфере записи 104 указывает, что число операций записи для хранения буфера равно его размеру.

За это время сигнатура или несколько сигнатур аккумулируются на основе вычисленных результатов, сохраняемых данных и их адресов в регистре сигнатуры 107.

После окончания выполнения инструкций на обнаружение одного из двух условий, описанных выше, накопленная сигнатура сравнивается с контрольным регистром 110, записанным ранее блоком 203. Если сигнатуры совпадают, значит ошибки не было, процесс продолжается с выполнения блока 304. Процессор переходит в состояние “0”, копируя содержимое файла-регистра 103 в контрольный регистр 109, содержимое счетчика команд 102 в резервный регистр 108 и инициализируя регистр сигнатуры 107. После этих шагов процессор находится в состоянии “0”. Если текущая сигнатура не совпадает с предыдущей, значит обнаружена ошибка (блок 305) и процессор переходит в состояние “2” (рис. 4).

Рис. 4. Состояние “2”

Рис. 5. Состояние “2” модифицированной архитектуры

Состояние “2” начинается с очистки буфера записи 104 (блок 401). Это делается в связи с тем, что была обнаружена ошибка и данные, хранимые в нем, могут быть неверными. Далее на внешнее устройство сообщается о том, что была обнаружена ошибка (блок 402). Это сообщение может быть использовано для обнаружения нестабильной среды или обнаружения устройства, приближающегося к отказу. Это состояние процессора инициализируется для повторного входа в состояние “0” без какой-либо информации от предварительного выполнения команд в состояниях “0” и “1”. Текущее значение счетчика команд 102 не копируется в контрольный регистр 108 (блок 403), текущее значение файл-регистра 103 не копируется в резервный регистр 109 (блок 404). Значение счетчика команд и файл-регистра восстанавливается из специальных регистров 108 и 109 соответственно, регистр сигнатуры 107 инициализируется (блок 405). Таким образом, состояние “0” будет начинаться в тех же начальных условиях, что и в предыдущий раз. После этого процессор переходит в состояние “0”.

Отказоустойчивый алгоритм, представленный на рис. 2–4, функционирует следующим образом. Блок контроля ошибок требует результаты, имеющие одни и те же сигнатуры. Содержимое буфера записи 104, оставшееся после предыдущего периода, выгружается во время первого прохождения следующего периода. Это позволяет буферу записи 104 осуществлять запись только достоверных результатов.

Архитектура 2. Внесем небольшие изменения в исходную архитектуру и алгоритм функционирования, а именно изменим алгоритм в состоянии “2”, как показано на рис. 5.

Отличие от начальной архитектуры только в блоке 505 и конечном переходе в состояние “1”. Таким образом, если переход в состояние “2” не был осуществлен (не было ошибки), то отличий между данной архитектурой и первоначальной не будет. Если же переход в состояние “2” был осуществлен (была ошибка), то очищается буфер записи 104 (блок 501). Как и в блоке 401, это сделано, потому что была обнаружена ошибка и данные, хранимые в нем, могут быть неверными. Далее на внешнее устройство сообщается об обнаружении ошибки (блок 502). Текущее значение счетчика команд 102 не копируется в контрольный регистр 108 (блок 503), текущее значение файл-регистра 103 не копируется в резервный регистр 109 (блок 504). Значение счетчика команд и файл-регистра восстанавливается из специальных регистров 108 и 109 соответственно, регистр сигнатуры 107 копируется в контрольный регистр 110 и инициализируется, (блок 505). Таким образом, состояние “1” будет начинаться в тех же начальных условиях, что и в предыдущий раз за исключением того, что в контрольном регистре сигнатуры 110 будет хранится новое значение сигнатуры. Далее осуществляется переход в состояние “1”. Как видно, внесенное изменение привело к тому, что блок контроля ошибок посчитает результат верным, если его сигнатура совпадает с сигнатурой предыдущего результата.

Архитектура 3. Архитектуру процессора можно дополнительно модифицировать путем добавления контрольного регистра сигнатуры 2 (блок 615 на рис. 6) и мажоритарного элемента (входит в блок 606) и соответствующим изменением состояний “1” и “2” так, как показано на рис. 7, 8.

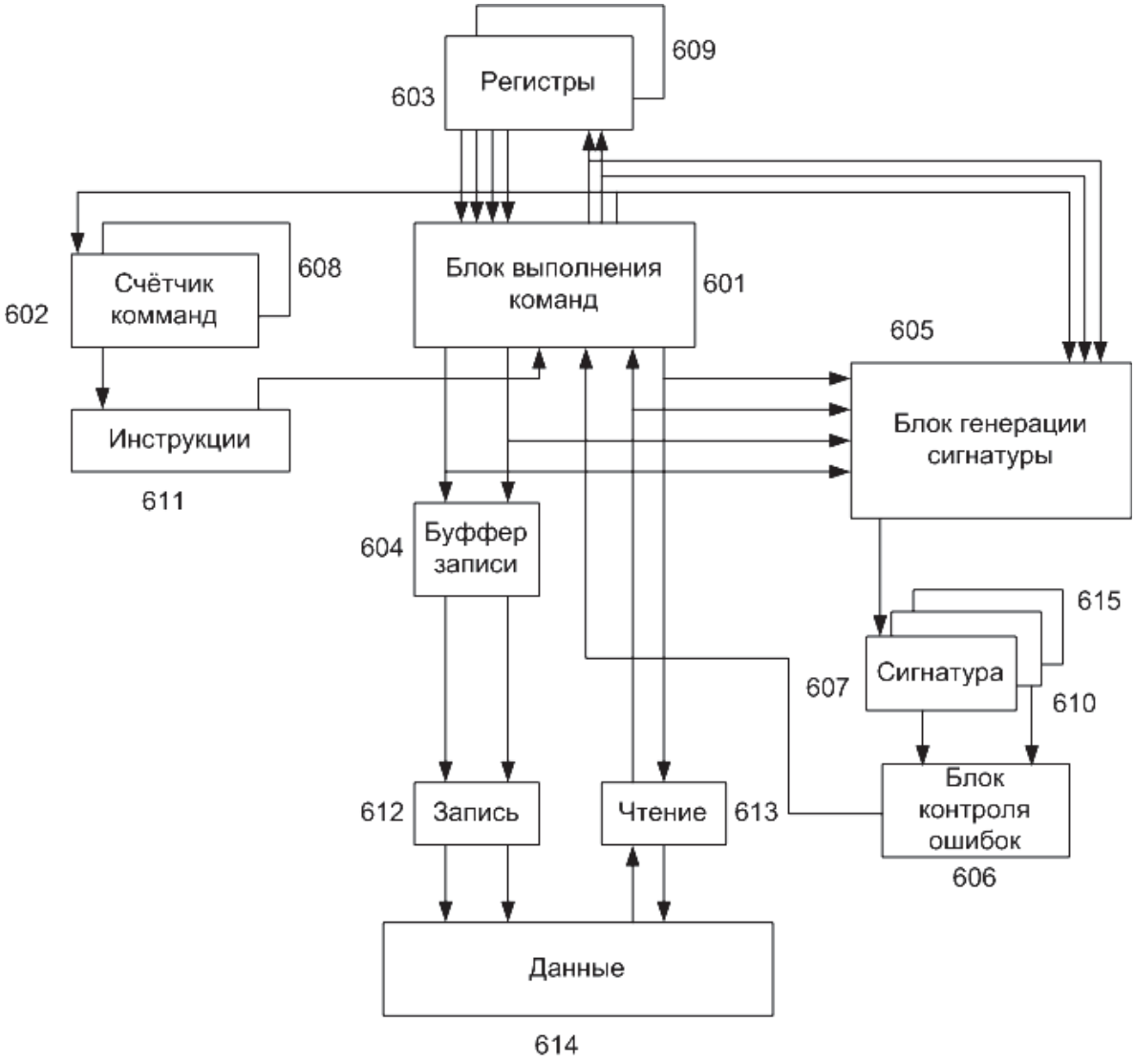

Рис. 6. Структура процессора с мажоритарным элементом

На рис. 7 представлен алгоритм состояния “1”. Блоки 701, 702 и 705 идентичны блокам 301, 302 и 305 (рис. 3) соответственно. После подсчета сигнатуры (блок 702) выполняется ее сравнение с сигнатурами, хранящимися в контрольных регистрах 610 и 615 посредством мажоритарного элемента (блок 703). В контрольном регистре 610 на момент сравнения хранится сигнатура предыдущего результата (вследствие работы блока 203 при первом сравнении и работы блока 805 при последующих сравнениях, которые возникнут только в случае обнаружения ошибки). Контрольный регистр 615 на момент первого сравнения инициализирован (в блоке 704), а при следующих сравнениях (если будет обнаружена ошибка) будет содержать сигнатуру, полученную на пред-предыдущем шаге (результат работы блока 805).

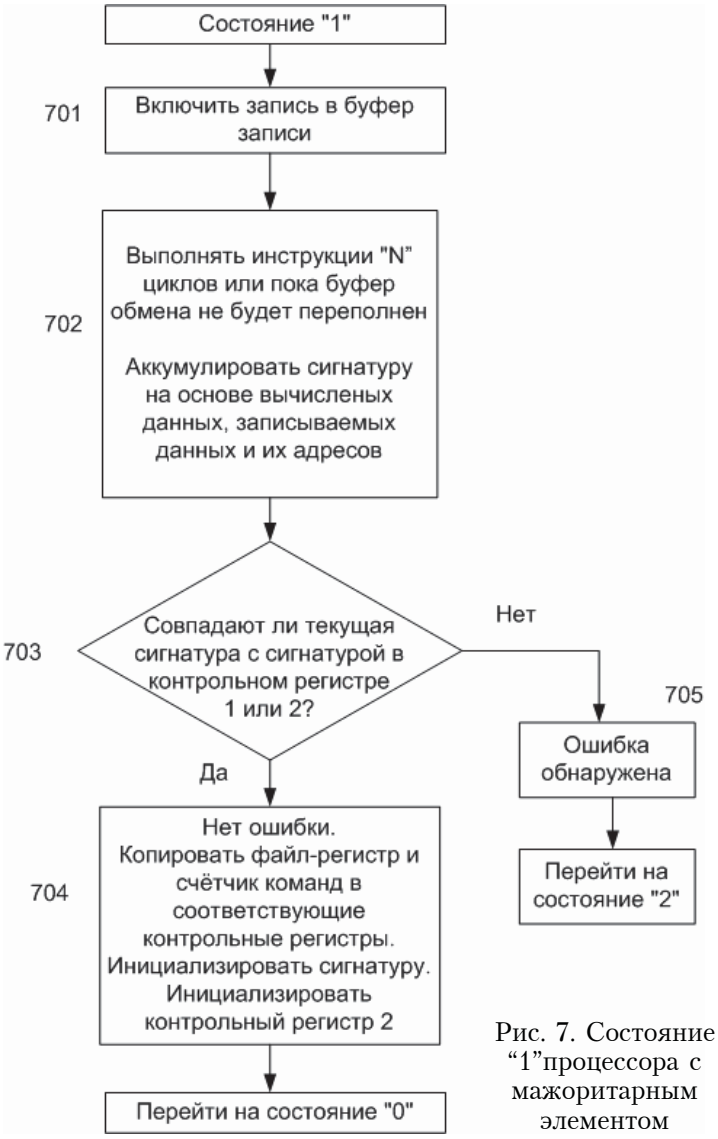

Рис. 7. Состояние “1”процессора с мажоритарным элементом

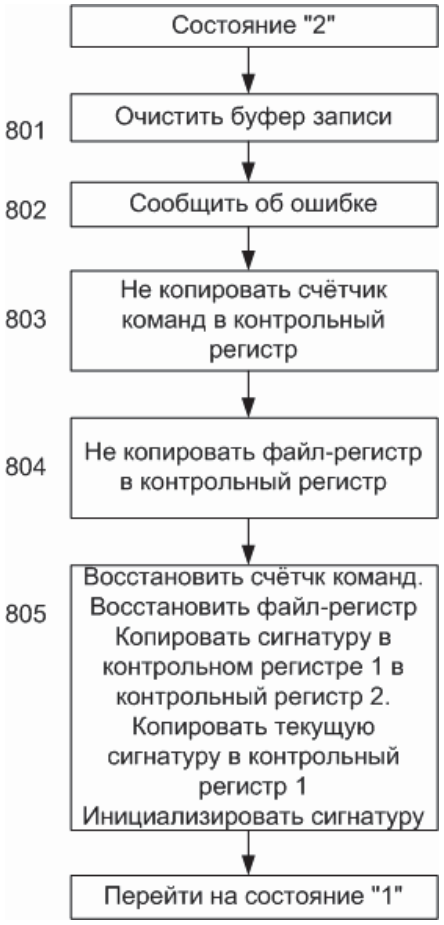

Рис. 8. Состояние “2”процессора с мажоритарным элементом

Таким образом, блок контроля ошибок посчитает результат правильным, если его сигнатура совпадает с одной из двух предыдущих сигнатур.

Сравнение архитектур. Определим вероятности получения ошибочного результата (ошибочной сигнатуры) для трех рассматриваемых архитектур процессоров.

Примем вероятность получения ошибочной сигнатуры на каждом цикле (шаге) равновероятной и равной p, тогда вероятность получения верной сигнатуры q = 1 - p. Выполнение команды менее чем за два цикла невозможно ни в одной из трех рассматриваемых архитектур. Обозначим через $P_i(n)$ вероятность выполнения команды на n-ом шаге в i-й архитектуре. Выполнение команды менее чем за 2 шага в рассматриваемых архитектурах невозможно и потому $P_i(n)=0$, при $n<2$ для любого i. Также отметим, что $P(2)= q^2$ для любой архитектуры, т.к. отличия в них наблюдаются только при обнаружении ошибки, а принятие результата на втором шаге свидетельствует об ее отсутствии.

Рассмотрим исходную архитектуру 1. В ней результат принимается только на четном числе циклов (2, 4, 6 ...). На каждой паре циклов (1 и 2, 3 и 4, ...) шанс

принять результат равен q^2, следовательно, шанс отвергнуть результат равен $(1-q^2)$. Для принятия результата на n-ом шаге необходимо что бы:

а) последние два результат были верными;

б) результат не был принят на предыдущих (n-2) шагах, т.е. был отвергнут (n-2)/2 раз.

Таким образом:

$$P_1(n=2k)=q^2(1-qq)^{k-1}.$$

Рассмотрим архитектуру 2. В ней результат принимается, если он совпадает с предыдущим полученным результатом. Для принятия результата на n-ом цикле необходимо что бы:

а) последние два результат были верными;

б) пред-предыдущий результат был неверен (иначе результат был бы принят на предыдущем цикле);

в) результат не был принят на (n-3)-х первых циклах.

Таким образом:

$$P_2(n) = p(p-1)^2(1-F(n-3)),$$

где $F(n)=\sum_{i=1}^{n}P(i)$ – функция распределения, равная вероятности получения правильного результата за n или менее циклов.

Рассмотрим архитектуру 3 с использованием мажоритарного элемента. В ней результат будет принят, если он совпадает с хотя бы одним из двух предыдущих. Заметим, что текущий результат не может быть равен одновременно двум предыдущим результатам, иначе результат был бы принят на предыдущем цикле. Пусть результат был принят на n-ом шаге и n>4. Предположим, что текущий результат равен предыдущему, тогда:

а) на n-ом и (n-1)-ом цикле был получен верный результат;

б) на (n-2)-ом и (n-3)-ем цикле были получены неверные результаты, иначе результат был бы принят на предыдущем цикле;

в) за первые (n-4) цикла результат принят не был.

Предположим, что текущий результат равен полученному на (n-2)-ом цикле:

а) на n-ом и (n-2)-ом цикле был получен верный результат;

б) на (n-1)-ом, (n-3)-ом и (n-4)-ом циклах были получены неверные результаты, иначе бы результат был бы принят на (n-2)-ом или (n-1)-ом цикле;

в) за первые (n-5) шагов результат принят не был.

Таким образом, вероятность принять результат на n-м шаге при n>4:

$$P_3(n) = (1 - F(n - 4))(pq)^2 + (1 - F(n - 5))p(pq)^2, \text{ или}$$
$$P_3(n)= (1-F(n - 4)+pF(n-5))(pq)^2,$$

где $F(n)=\sum_{i=1}^{n}P(i)$ – функция распределения, равная вероятности получения правильного результата за n или менее циклов.

Среднее количество циклов (фактически инструкций), за которое будет выполнена одиночная инструкция для любой архитектуры, определяется матема-

тическим ожиданием функции Pi(n) соответствующей архитектуры и напрямую зависит от вероятности ошибки p: $N_{cp} = \sum_{i=1}^{\infty} iP(i)$.

На рис. 9, 10 приведены зависимости среднего числа выполняемых циклов (соответственно инструкций) от вероятности ошибки p. Исходной архитектуре соответствует графике 1, модифицированной – 2, архитектуре с использованием мажоритарного элемента – 3.

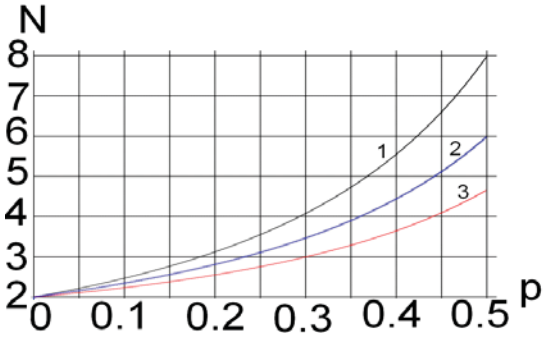

Рис. 9. Среднее число циклов при p<0,5

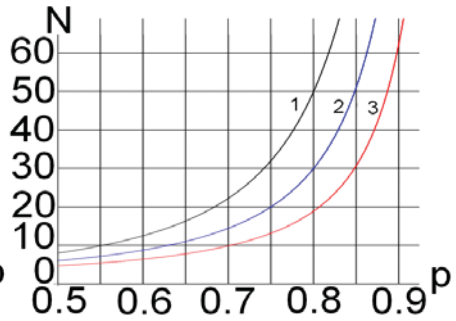

Рис. 10. Среднее число циклов при p>0,5

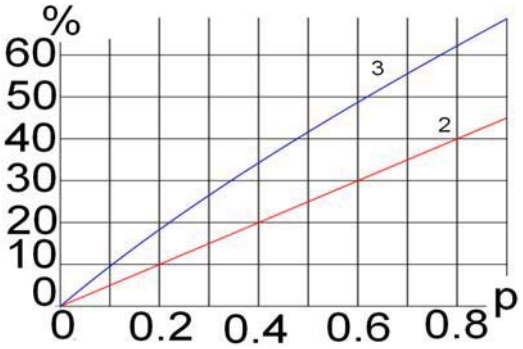

Рис. 11. Относительное уменьшение циклов (инструкций)

Как видно из рис. 9, 10, при любой вероятности ошибки p для архитектуры 3 с мажоритарным элементом среднее количество циклов (выполнения инструкций) для получения результата минимально, а для исходной архитектуры 1 — максимально. На рис. 11 приведены относительные уменьшения среднего количества циклов (инструкций) для модифицированной архитектуры (график 2) и архитектуры с мажоритарным элементом относительно исходной архитектуры 1.

Очевидно, что для повышения скорости работы системы целесообразно использовать архитектуру 3 с мажоритарным элементом, однако при ее использовании необходимо наличие дополнительного регистра, а так же, в случае ошибки, необходимо дважды копировать сигнатуру, что не надо делать при других типах архитектуры.

Выводы

В роботе проанализированы принципы построения помехозащищенных архитектур процессоров для современных вычислительных систем. Описаны функ-

ционирование и особенности модернизации для трех архитектур, показаны их преимущества и недостатки, приведены зависимости скорости работы (количества исполняемых инструкций) от вероятности ошибки вычислений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Затуливетер Ю.С.* Разработка и исследование методов повышения надежности распределенных вычислений / Ю.С. Затуливетер, Е.А. Фищенко, И.А. Ходаковский // Надежность. – 2009. – Вып. 1. – С. 42–49.

2. *Матушевский В.В.* Вычислительные системы : учебное пособие / В.В. Матушевский, Ю.А. Гунченко. – Одесса : ВМВ, 2011. – 204 с.

3. DonaldE. Steiss, Single event upset tolerant microprocessor architecture [Електронний ресурс]. – Режим доступу : http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2 Fsrchnum.htm&r=1&f=G&l=50&s1=6,571,363.PN.&OS=PN/6,571,363&RS=PN/6,571,363.

4. *Гнеденко Б.В.* Курс теории вероятности : Издание шестое / Б.В. Гнеденко. – Москва : Наука. – 448 с.

5. Пат 67752 Україна, МПК(2006.01) G06F 12/08. Пристрій підвищення завадостійкості систем з програмним управлінням / Гунченко Ю.О., Мартинюк С.М., Лєнков С.В., Омельченко О.С., Купрацевич А.В. ; заявник і патентовласник Одеський національний університет ім. І.І. Мечникова. – № u 201107423 ; заявл. 14.06.2011 ; опубл. 12.03.2012, Бюл. № 5.

6. *Омельченко А.С.* Архитектура процессора для отказоустойчивых вычислительных систем / А.С. Омельченко, И.В. Мигов, Ю.А. Гунченко // Восьма Регіональна конференція студентів і молодих науковців. – 2011. – С. 17–18.

7. Пат 76984 Україна, МПК (2006.01) G06F 11/27. Відмовостійкий процесорний пристрій з підвищеною швидкодією / Гунченко Ю.О., Лєнков С.В., Кобозєва А.А., Мартинюк С.М., Борисенко І.І. ; заявник і патентовласник Одеський національний політехнічний університет. – № u 2012 07958 ; заявл. 27.06.2012 ; опубл. 25.01.2013, Бюл. № 2.

8. *Гунченко Ю.О.* Завадостійкий процесорний пристрій з підвищеною швидкодією / Ю.О. Гунченко, А.С. Омельченко, О.А. Пенко // Тези доповідей Третьої Міжнародної науково-практичної конференції “Методи та засоби кодування, захисту й ущільнення інформації”. – 2011. – С. 42.

Отримано 01.04.2014